

DOCUMENT RESUME

ED 124 130

IR 003 518

AUTHOR Avner, Elaine
TITLE PLATO User's Memo, Number One: Summary of TUTOR
Commands and System Variables. Third Edition.
INSTITUTION Illinois Univ., Urbana. Computer-Based Education
Lab.
SPONS AGENCY Illinois Univ., Urbana.; National Science Foundation,
Washington, D.C.
PUB DATE Nov 75
GRANT USNSF-C-723
NOTE 96p.
AVAILABLE FROM PLATO Publications, Computer-based Education Research
Laboratory, 252 Engineering Research Laboratory,
University of Illinois, Urbana, Illinois 61801
(\$2.95, prepayment required)

EDRS PRICE MF-\$0.83 HC-\$4.67 Plus Postage.
DESCRIPTORS *Computer Assisted Instruction; Higher Education;
Instructional Systems; *Manuals; *Programming
Languages
IDENTIFIERS PLATO IV; Programmed Logic for Automated Teaching
Operations; *TUTOR

ABSTRACT

Requiring a knowledge of the PLATO system and a command of the TUTOR language, this manual describes the format and the restrictions for the following TUTOR operations: (1) calculations; (2) data keeping; (3) judging responses; (4) presenting; (5) routing; and (6) sequencing. Within each section, relevant commands are defined, and restrictions are established. The appendixes include a list of the limits associated with each command and alphabetical listings of systems variables and of commands.
(EMH)

* Documents acquired by ERIC include many informal unpublished *
* materials not available from other sources. ERIC makes every effort *
* to obtain the best copy available. Nevertheless, items of marginal *
* reproducibility are often encountered and this affects the quality *
* of the microfiche and hardcopy reproductions ERIC makes available *
* via the ERIC Document Reproduction Service (EDRS). EDRS is not *
* responsible for the quality of the original document. Reproductions *
* supplied by EDRS are the best that can be made from the original. *

PLATO USER'S MEMO

Number 1

Third Edition

Summary of TUTOR Commands
and System Variables

Elaine Avner

November, 1975

U.S. DEPARTMENT OF HEALTH,
EDUCATION & WELFARE
NATIONAL INSTITUTE OF
EDUCATION

THIS DOCUMENT HAS BEEN REPRODUCED EXACTLY AS RECEIVED FROM THE PERSON OR ORGANIZATION ORIGINATING IT. POINTS OF VIEW OR OPINIONS STATED DO NOT NECESSARILY REPRESENT OFFICIAL NATIONAL INSTITUTE OF EDUCATION POSITION OR POLICY.

CERL



Computer-based Education Research Laboratory

University of Illinois

Urbana Illinois

PERMISSION TO REPRODUCE THIS COPY-
RIGHTED MATERIAL HAS BEEN GRANTED BY

Computer-based Education

Research Lab., Univ. of Illinois
TO ERIC AND ORGANIZATIONS OPERATING
UNDER AGREEMENTS WITH THE NATIONAL IN-
STITUTE OF EDUCATION. FURTHER REPRO-
DUCTION OUTSIDE THE ERIC SYSTEM RE-
QUIRES PERMISSION OF THE COPYRIGHT
OWNER

Copyright © November 1975
by Board of Trustees, University of Illinois

First Edition May 1974
Second Edition June 1975
Third Edition November 1975

All rights reserved. No part of this book may be reproduced in any form or by any means without permission in writing from the author.

This manuscript was prepared with partial support from the National Science Foundation (USNSF C-723) and the University of Illinois at Urbana-Champaign.

ACKNOWLEDGMENT

Many PLATO users gave helpful suggestions on past editions of this book: Allen Avner, Richard Blomme, Robert Bohn, George Friedman, James Ghesquiere, John Gilpin, William Golden, Bruce Sherwood, Judy Sherwood, and Paul Tenczar.

Sheila Knisley typed the manuscript. Roy Lipschutz did the drafting work, and Wayne Wilson prepared the keyset diagram.

This summary is intended for the experienced author who needs a quick reference for the form of a tag and for some of the restrictions on commands. It is not designed to teach the fundamentals of TUTOR, nor to discuss TUTOR states, answer-judging, sequencing, etc. For such information authors may refer to "aids" and to The TUTOR Language by Bruce Arne Sherwood.

Each command includes a brief description of its purpose and a description of the tag. The standard form is

command brief description

command *description of tag* (any explanatory comments)

or

command actual tag

Note: Additional comments about this command

NOTE: General comments about groups of commands

The commands are grouped into six categories: calculating (C), data keeping (D), judging (J), presenting (P), routing (R), and sequencing (S). Each section is identified by the page number (e.g., C1 or P5) and by the mark at the edge of the page. By matching the marks at the edges of the index pages with the marks along the edges of the book, a user may quickly access the desired section.

Modifications to this book required by changes in the TUTOR language are contained in lesson "aids," option "Changes to 'Summary of TUTOR Commands'" (also reached via DATA, summary). These modifications, along with other notes of interest, may be inserted in the spaces which have been provided between entries and on the additional pages at the end of each section.

Contents

	Page	
Abbreviations	i	
Classification of commands and system variables	ii to vii	
CALCULATING		
Basic calculating	C1	
System operations and functions	C4	
Random numbers	C7	
Information	C8	
Bit and character manipulation	C9	C
Blocks of data	C11	
System variables for calculating	C17	
DATA KEEPING		
Requesting data	D1	
Classifying data	D2	
Transferring data	D3	D
Student sign-on and sign-off	D6	
System variables for data keeping	D7	
JUDGING		
Prepare response	J1	
Establish lists	J3	
Modify judging copy of response	J4	
Modify judging procedure	J5	
Store judging copy of response	J7	
Match judging copy of response	J8	J
Specify judgment unconditionally	J12	
Reference other units which may contain judging commands	J13	
Alter judgment	J14	
Alter feedback	J15	
System variables for judging	J16	
PRESENTING		
Text	P1	
Graphics	P6	
Drawing graphs	P8	
Non-screen	P13	P
Special display	P14	
System variables for presenting	P16	
ROUTING		
Router lesson	R1	
System router	R2	R
System variables for routing	R3	
SEQUENCING		
Basic sequencing	S1	
Automatic sequencing	S2	
Key-initiated sequencing	S5	
Timing	S7	S
Lesson connections and sections	S9	
Lesson lists	S11	
Lesson annotation and debugging	S13	
System variables for sequencing	S16	
Appendix - Some limits associated with commands	A1	A
Alphabetical index		
System variables	I1	I
Commands	I2	
Keyset		

Abbreviations and Notes

Below are listed the abbreviations used in the descriptions.

abbreviation	definition
arg	argument or tag entry
(b)	blank tag
coarse	character grid coordinates
CM	central memory
CPU	central processing unit
disk	rotating magnetic disk storage unit
ECS	extended core storage
expr	mathematical expression
num	number of
finex, finey	fine grid coordinates
string	character string
var	variable
vars	variables

In conditional statements and in statements where a variable is set, suffixes m, $\emptyset$, 1, 2, etc., denote the minus condition, $\emptyset$ condition, 1 condition, 2 condition, etc., e.g.,

```
match  var, word $\emptyset$ , word1, word2
do      expr, namem, name $\emptyset$ , name1, name2
```

In conditional statements the conditional expression is rounded (not truncated) to the nearest integer.

Except where indicated otherwise, use of v-variables is recommended.

Generally whenever a tag entry may be a number, a mathematical expression will also be accepted.

Command names are enclosed in dashes when they are referred to in the descriptions, e.g., -next-. Key names are capitalized, e.g., NEXT.

Commands labeled "non-executable" are active only when the lesson is being condensed.

When variables are used in the tag of certain commands which require names in the tag, e.g., -area-, the variable must be enclosed in parentheses to indicate that the information needed is the contents of the variable and not a character string; e.g., -area (v3)- means the area whose name is contained in variable v3, while -area v3- means the area whose name is v3.

In the tags of some commands both commas and semicolons are used as separators, e.g., -draw-, -from-, -transfr-, conditional -jumpout-, etc. Correct placement of the separators is critical.

CALCULATING

Basic calculating C1

define
calc
calcc
calcs
addl
subl
zero
set

Random numbers C7

randu
setperm
randp
remove
modperm

Information C8

clock
date
day
name
course
compute

Bit and character manipulation C9

pack
find
findall
search
itoa
move

Blocks of data C11

block
transfr
common
comload
abort
initial
storage
stoload
dataset
datain
dataout
reserve
release
backgnd
foregnd

System variables for calculating C17

lcommon
lstoraq
zbpq
zbpw
zcpw
zdsrecs
zdsopr

DATA KEEPING

Requesting data D1

dataon
dataoff

Student sign-on and sign-off D6

restart
finish

Classifying data D2

area
output
output1

System variables for data keeping D7

aarea
aarrow
ahelp
ahelpn
aok
aokist
asno
aterm
atermn
atime
auno
dataon

Transferring data D3

readset
readd
readr

JUDGING

Prepare response J1

erased
force
edit
arrow
long
jkey
copy
endarrow

Establish lists J3

list
endings
vocabs
vocab

Modify judging copy
of response J4

bump
put
putd
putv
close
loada

Modify judging procedure J5

specs

Store response J7

store
storeu
storen
storea
open

Match response J8

match
answer
wrong
answerc
wrongc
concept
miscon
exact
exactc
exactv
ansv
wrongv
ansu
wrongu
touch
touchw
or
ans

Specify judgment
unconditionally J12

ok
no
ignore

Reference other units
which may contain
judging commands J13

join
iarrow

Alter judgment J14

judge

Alter feedback J15

okword
noword

System variables
for judging J16

judging in general

anscnt
ansok
jcount
judged
key
ntries

verbal responses

capital
entire
extra
order
phrase
spell
wcount

numerical response

formok
opcnt
varent

PRESENTING

Text P1

at
atnm
write
writec
show
showz
showt
showe
showo
showa
erase
mode
color
size
rotate
inhibit
delay
char
plot

Drawing graphs P8

origin
axes
bounds
frame
scalex
scaley
lscalex
lscaley
labelx
labley
markx
marky
locate
graph
gdraw
polar
vector
vbar
delta
funct

Non-screen P13

slide
audio
play
record
enable
disable
ext

Special display P14

tabset
micro
charset
chartst
lineset
altfont

System variables for presenting P16

Graphics P6

dot
draw
rdraw
circle
circleb
window

mode
size
where
wherex
wherey

ROUTING

Router lesson R1

route
routvar
allow

System router R2

lesson
score

System variables for routing R3

errtype
ldone
lscore
rcallow
router
rstartl
rstartu
rvallo

SEQUENCING

Basic sequencing S1

unit
entry

Automatic sequencing S2

jump
goto
do
join
exit
iferror
imain
branch
doto

Key-initiated sequencing S5

next
next1
back
back1
stop
nextnow
help
help1
data
data1
lab
lab1
helpop
help1op
dataop
data1op
labop
lab1op
term
base
end

Timing S7

group
pause
keytype
time
timer
time!
press
catchup
return
cpulim

Lesson connections
and sections S9

cstart
cstop
cstop*
use
jumpout
from
lessin
in

Lesson lists S11

leslist
addlst
removl
lname
findl

Lesson annotation
and debugging S13

*
c
\$\$
change
step
*list

System variables
for sequencing S16

args
backout
baseu
clock
fromnum
key
lessnum
llesson
mainu
mallot
muse
nhelpop
proctim
ptime
sitenam
station
tactive
user
usersin
zreturn
zwpb

CALCULATING

Basic calculating

define (non-executable) permits author to rename variables, define mathematical functions, and define constants for a lesson and to specify those available for student use; defined variables must physically precede any reference to the variables in the lesson

for example:

```
define setname
  name1=v1,name2=n2,name3=85,name4=2(name1+name3),...
  func(x,y,...)=some function of x, y, etc., where x, y,
  etc., are not already defined, although function on
  right of equal sign may contain previously defined names
  (up to 6 arguments are permitted)
```

The following definitions allow use of segmented variables.

```
segment,name=starting var,num bits per byte,s (optional)
segment,vertical,name=starting var,starting bit position,
  num bits per byte,s (optional)
(Starting variable address and byte size cannot be
variables, and byte size is from 1 to 59. Starting
variable may be a v-variable. If last argument is
included, negative as well as positive integers may
be stored. In vertical segment, starting bit position
may be from 1 to 60.)
```

The following definitions allow use of arrays.

```
array,name(size)=starting var (number of variables
  required equals size)
array,name(num rows,num columns)=starting var
  (number of variables equals rows×columns)
array,name(first element;last element)=starting var
array,name(first row element,first column element;last
  row element,last column element)=starting var
  (up to 255 elements are permitted in an array)
```

```
define student
  all defines necessary for student responses, including units
  units,unit1,unit2,... (maximum of 10 units; such as gram,
  meter,...)
  (The define set "student" may also include abbreviations
  and equivalences involving these units.)
```

(See -storeu-, -ansu-, and -wrongu- for applications.)

(Continued on next page)

```

define seta,setb (merges define sets named)
define purge,setname (discards define set named)
define purge (discards all define sets except "student")

```

Note: Up to 400 definitions are permitted in a define set, less if definitions are long. Defined variable names and names of define sets cannot exceed 7 characters, cannot contain mathematical operators, and must start with a letter.

calc assigns value of expression on right side of assign arrow to variable on left side, or packs up to 10 characters into an integer variable

```

calc var <= expr
calc var <= "string" (right-justified; use n-variable)
calc var <= 'string' (left-justified; use n-variable)
calc array <= expr (includes standard arithmetic operations,
                    bit operations, logical operations, and array functions)
calc var <= arrayname1.arrayname2
calc arrayname1 <= arrayname2.arrayname3

```

Note: See section on SEQUENCING, Automatic sequencing for uses of -branch- and -doto-, which are calc-type commands which allow branching within a unit.

calcc does one of several calculations depending on the rounded value of an expression

```
calcc expr, var1 <= exprm, var2 <= expr0, var3 <= expr1, var4 <= expr3, ...
```

calcs sets a variable to one of several values depending on the rounded value of an expression

```
calcs expr, var <= exprm, expr0, expr1, expr2, expr4, ...
```

NOTE: Up to 61 calculations may be performed with a -calcc- or -calcs-. A blank tag entry (,) means no calculation is done for corresponding value of the conditional expression.

add1 adds 1 to variable specified in tag

```
add1 var
```

sub1 subtracts $\frac{1}{\text{tag}}$ from variable specified in tag
 sub1 *var*

zero sets to zero a single variable or a set of consecutive variables
 zero *var*
 zero *starting var, num vars*

set sets values of consecutive variables starting at the specified
 variable, or sets values of matrix elements starting at the
 specified element (starts at the first element if no element is
 specified); cannot be used to set segmented variables

set *starting var &value1,value2,value3*
 set *arrayname &value1,value2,value3*
 set *arrayname(row,column) &value1,value2,value3*

Operations and symbols used with calculations

addition +

subtraction -

multiplication $\times$ or $*$ (implied multiplication is also permitted, e.g., 5a)

division $:$ or $/$

parentheses, brackets (), [], { }

exponentiation with superscript or shift superscript, e.g., a^4

assignment of a value to a variable $\Leftarrow$

$\pi = \text{pi} = 3.14159\dots$

$^\circ$ = degree sign; indicates a number is in degrees, e.g., 30° ; number $\times 1^\circ$ converts number to radians; number $\div 1^\circ$ converts number to degrees

Address of a variable may be an expression; e.g., $v(\text{expr})$ is permitted, where expr is rounded to the nearest integer.

Precedence of operations (in brief)

operations within parentheses

exponentiation

multiplication

division

addition and subtraction

In general with anything but very simple expressions, parentheses should be used freely.

NOTE: The computer has approximately 14-digit accuracy.
 Values of v-variables may range from about $\pm 10^{-293}$ to $\pm 10^{+322}$.
 Values of n-variables may range from about -10^{17} to $+10^{17}$.
 However, multiplication and division of large integer values may give erroneous results because of limitations on integer arithmetic.

System functions (argument may be an expression)

abs(x)	absolute value of x	
arctan(x)	inverse tangent, result in radians, range $-\pi/2$ to $+\pi/2$; for result in degrees, use $\text{arctan}(x)/1^\circ$	
cos(x)	cosine of x, x in radians; $\cos(x^\circ)$ when x is in degrees	
exp(x)	e^x	
frac(x)	fractional part of x	x is first rounded to the nearest integer if x is within 10^{-9} of the integer for $ x < 10^5$ OR within $10^{-14} x $ of the integer for $ x > 10^5$ (approximately)
int(x)		
log(x)	common logarithm of x	
ln(x)	natural logarithm of x	
round(x)	rounded value of x	
sign(x)	returns -1 for $x < 0$; 0 for $x = 0$; +1 for $x > 0$	
sin(x)	sine of x, x in radians; $\sin(x^\circ)$ when x is in degrees	
sqrt(x)	square root of x	
varloc(x)	address of variable x (x may be a student variable, central memory variable, or defined variable)	

Logical operations and functions (logical "true" is -1; logical "false" is 0)

x=y	equal to	$\left. \begin{array}{l} \text{equality is "true" if} \\ x-y < 10^{-9} \text{ for } x < 10^8 \text{ OR} \\ \text{if } x-y < 10^{-11} x \text{ for } x > 10^8 \\ \text{(approximately)} \end{array} \right\}$
x≠y	not equal to	
x≤y	less than or equal to	
x≥y	greater than or equal to	
x<y	less than	
x>y	greater than	
x&y	logical "and"; result is "true" only if both x and y are "true"	
x y	logical "or"; result is "true" if either x or y or both are "true"	
not(x)	reverse of truth value of x: if $x=0$, $\text{not}(x)=-1$; if $x \neq 0$, $\text{not}(x)=0$	

Bit operations and functions (use with n-variables)

x\$ars\$y	shifts x to the right by y bit positions	
x\$cls\$y	shifts x to the left (circularly) by y bit positions	
x\$mask\$y	sets bits where bits are set in both x and y	
x\$union\$y	sets bits where bits are set in either x or y or both	
x\$dif\$y	sets bits where bits are set in either x or y but not both	
bitcnt(x)	number of bits set in x	
comp(x)	complement of x (bit reversal)	
lmask(x)	left-justified octal number of x bits	} x ranges from 0 to "zbpw"
rmask(x)	right-justified octal number of x bits	

Array functions

And(x) "true" (=1) if all elements of array x are "true"
Max(x) largest element in array x
Min(x) smallest element in array x
Or(x) "true" (=1) if any element of array x is "true"
Prod(x) product of all elements in array x
Rev(x) reverse of array x, i.e., last element is now first, etc.
Sum(x) sum of all elements in array x
Transp(x) transpose of array x, i.e., rows and columns are interchanged

NOTE: Because of the finite accuracy of any computer, rounding occurs with operations with fractional values (v-variables), giving results which may be off by only one or two bits but which can lead to serious errors. The tolerances indicated above are designed to avoid such problems by ignoring these least significant bits. However, there is no general solution to this inherent problem, and users must design checks for specific applications.

Random numbers

randu selects a random number, sampled with replacement, and places it in specified variable

randu *var, maximum integer* (provides integers from 1 to maximum)
randu *var* (provides numbers from 0 to 1; if the tag is an n-variable, value 0 or 1 is returned)

NOTE: The next four commands are generally used together to provide random numbers without replacement.

setperm sets up two lists of integers from 1 to value specified in tag for sampling without replacement, one a working copy of the list and the other an inactive but alterable copy of the list

setperm *maximum integer in list* ($0 \leq \text{maximum} \leq 120$)
setperm *length of list, starting var containing elements in list*
 (for lists > 120 and for special purpose sampling;
 number of variables required is $2 + \text{int}[(\text{length} - 1) / 60]$;
 length of the list is from 0 to 30000)

randp selects a random integer without replacement from working copy of list set up by **-setperm-** and places it in specified variable

randp *var*
randp *var where value is stored, starting var containing elements in list*

Note: With either form of **-randp-** value returned is 0 if working copy of list is exhausted.

remove removes specified value from the inactive copy of the list set up by **-setperm-**

remove *integer to be removed* (list length ≤ 120)
remove *integer to be removed, starting var of copy of list*

modperm (no tag) replaces working list with current copy of inactive list from which integers may have been removed by **-remove-** command; used with list length ≤ 120 only; to simulate **-modperm-** for list length > 120 use **-block-** or **-transfr-**

Information

clock puts character string in tag variable giving time on 24-hour clock in format (space) hour.minute.second. (see also system variable "clock")

clock *var where string is stored* (use -showa- to display)

date puts character string in tag variable for current date in format (space) month number/day/last two digits of year (space)

date *var where string is stored* (use -showa- to display)

day places in tag variable the number of days elapsed since midnight December 31, 1972, to nearest 10^{-6} day (approximately .1 second)

day *var where value is stored*

name places sign-on name of user (up to 18 characters) left-justified in two variables starting at tag variable with octal zero-fill in unused positions

name *starting var* (requires two consecutive variables)

course places sign-on course of user left-justified in tag variable, up to 8 characters with octal zero-fill in unused positions

course *var*

compute compiles specified character string, executes, and stores result in specified variable; sets pointer to compiled code

compute *var where result is stored, starting var of string, num characters in string, var which is pointer to compiled code* (invalid expression does not compile; once compiled, no recompilation is done unless pointer is zeroed)

Note: Character string may be stored with -storea- or -pack- and may contain up to 99 characters.

Bit and character manipulation

pack packs a character string, left-justified, starting in the specified variable with octal zero-fill in unused positions

pack *var where character count is stored (optional), starting var where string is stored, string*

Note: Use *n*-variable(s) for packing character string if subsequent comparison for equality with another string is done. (If character string is packed for other purposes, *v*-variables are acceptable.) Segmented variables cannot be used with *-pack-*.

find scans each variable in a set of consecutive variables for the first variable containing the specified bit pattern (see NOTE below)

find *arg1, arg2, arg3, arg4, arg5, arg6*

arg1 = object bit pattern

arg2 = starting variable in list

arg3 = number of variables in list

arg4 = variable for storing relative found location

arg5 = increment between variables (optional)

arg6 = mask (optional)

findall scans each variable in a set of consecutive variables for the variables containing the specified bit pattern (see NOTE below)

findall *arg1, arg2, arg3, arg4, arg5, arg6, arg7*

arg1 = object bit pattern

arg2 = starting variable in list

arg3 = number of variables in list

arg4 = variable for storing found count

arg5 = number of following variables for storing relative found locations

arg6 = increment between variables (optional)

arg7 = mask (optional)

NOTE: Use *n*-variables with *-find-* and *-findall-*. Segmented variables may not be used. Increment, if omitted, is 1. Negative increment causes a backward scan from the last variable in the list. If the mask is omitted, the entire variable is compared with the object bit pattern. If a mask is used, the increment must be specified, even if it is 1. Relative position of the first variable is 0, of the second variable, 1, etc. With *-find-*, if the bit pattern is not found, the found location is -1. With *-findall-*, if the bit pattern is not found, the count is 0 and the first following variable is -1.

search scans a character string for a specified object character string

search *arg1, arg2, arg3, arg4, arg5, arg6*

arg1 = object string (left-justified)
arg2 = number of characters in object string (≤ 10 characters)
arg3 = starting variable of string to search
arg4 = number of characters in string to search
arg5 = relative character position at which to start search
arg6 = variable for storing relative character position
 of first occurrence of object string

search *arg1, arg2, arg3, arg4, arg5, arg6, arg7*

arg1 = object string (left-justified)
arg2 = number of characters in object string (≤ 10 characters)
arg3 = starting variable of string to search
arg4 = number of characters in string to search
arg5 = relative character position at which to start search
arg6 = variable for storing found count
arg7 = number of following variables for storing relative
 character positions of object string

Note: In both versions of **-search-** the relative found location is -1 if the object string is not found. In the second version the count is 0 if the string is not found. Relative position of the first character is 1, of the second character, 2, etc.

itoa converts an integer to a character string, left-justified with octal-zero fill in unused positions

itoa *var where integer is stored, starting var for storing string, var for storing num characters (optional)*

Note: Non-integer values are rounded to the nearest integer before conversion to a character string

move moves character(s) from a specified character position in a character string to a specified position in another character string

move *from starting var, from starting position, to starting var, to starting position, num characters (optional)*

move *'string', from starting position, to starting var, to starting position, num characters (optional)*

Note: If not specified, number of characters is 1. Maximum number of characters is 5000. Positions need not be ≤ 10 since character strings are being considered.

Blocks of data

block copies a set of consecutive student variables (v, n) or central memory variables (vc, nc) into another set of consecutive variables

block *from starting var, to starting var, num vars*

transfr transfers data between v- and n-variables (student variables), vc- and nc-variables (central-memory variables), ECS common, ECS storage, or vr- and nr-variables (router variables) (-comload- and/or -stoload- must be in effect with central memory variables)

transfr *from starting location; to starting location; length*
(general form)

Any of the following may be used in the first two arguments of the tag:

student var
common, ECS common location or c, ECS common location
storage, ECS storage location or s, ECS storage location
CM vc- or nc- var
router, ECS router common location (when placed in "from" position, router lesson must contain -allow read-; when placed in "to" position, router lesson must contain -allow write-)
routvars, ECS router var location (may be placed only in "from" position; router lesson must contain -allow read rvars-)
router var (legal only when -transfr- command is in the router lesson)

for example:

transfr v1;c,23;10 (transfers variables v1 through v10 to ECS common location 23 to 32)
transfr v6;vc51;9 (transfers variables v6 through v14 to vc51 through vc59)

Note: Limit to length which may be transferred is set by:
tag of -common- or -storage- (when referencing ECS) OR
length of -comload- or -stoload- (when referencing CM variables) OR
150 (when referencing student variables) OR
tag of -routvar- command (when referencing router variables)

NOTE: For this type of operation, -block- and -transfr- are very fast.

common (non-executable) sets up storage space which is accessible to all students in a particular lesson; in central memory, the variables are referenced by *vc* and *nc*; common codewords must match when common blocks are in a different lesson from the *-common-* statement

common *length of temporary common* (maximum length of 1500; variable tag is not permitted)

common *lesson name, block(s) name, length of permanent common* (for length ≤ 1500 , loading and unloading are automatic unless altered by *-comload-*, *-inhibit unload-*, or optional tags "no load" or "read only"; for length > 1500 , *-comload-* must be used; maximum length of permanent common is 8050)

Note: Either or both of the following optional arguments may be added to the tag of the *-common-* command:

no load (cancels automatic loading of common from ECS to CM)

read only (prevents transfer of common from CM to ECS)

The following optional argument may be used only with permanent common:

checkpt (causes common to be returned to disk approximately every 8 minutes)

comload provides automatic loading and unloading of permanent common between central memory and ECS during each time slice; required if common length exceeds 1500 (see *-inhibit unload-*)

comload *starting CM var (vc or nc), ECS common location, num vars* (maximum of 1500 variables)

comload *(b)* (uploads CM variables and turns off further automatic loading until another *-comload-* command is executed)

for example,

comload *vc22, 10, 8* (transfers *vc22* to *vc29* from and to ECS common locations 10 to 17)

abort prevents return of information to the disk

abort *common*

abort *record* (with student record; sets "user" to "sabort")

abort *autocheck* (with student record; prevents automatic return of records every 8 minutes; sets "user" to "snockpt")

abort *(b)* (clears previous *-abort-* settings)

initial specifies a unit to be executed by the first user to encounter this command in a lesson when the lesson or common is first brought into ECS

initial common, *name* (not executed if common is in another lesson which is already in ECS)

initial lesson, *name*

storage (non-executable) sets up storage space which serves as temporary extension of student variables; in central memory, the variables are referenced by *vc* and *nc*; they are not saved in student records and are zeroed during jumpout to another lesson (for exception see **-inhibit dropstor-**)

storage *length of storage* (maximum length of 1500)

stoload, similar to **-comload-** but refers to storage; required for any length storage to access central memory variables (see **-inhibit unload-**)

stoload *starting CM var (vc or nc), ECS storage location, num vars*
(maximum of 1500 variables)

stoload (b) (turns off automatic loading and unloading of storage)

NOTE: When using both **-comload-** and **-stoload-**, care must be exercised that the addresses of the central memory variables into which common has been loaded do not overlap with the addresses into which storage has been loaded.

dataset establishes connection with special type of file which permits disk records to be read and written in (via **-datain-** and **-dataout-**); dataset file has maximum size of about 40,000 words; record size may be from 64 to 512 words, with default of 322

dataset name (common codewords on lesson and on dataset must match)
dataset name, expr, 'codeword' (optional) (third argument is required if common codewords on lesson and on dataset do not match; for read and write access, *expr*=-1 and *codeword* is dataset common codeword; for read only access, *expr*≠-1 and *codeword* is either dataset common codeword or inspect codeword)

Note: Variable first or third argument must be enclosed in parentheses; if the third argument is variable, quote marks are omitted.

zreturn = -2 if connection to the dataset is successful
 (tag codeword match)
 = -1 if connection to the dataset is successful
 (common codeword match)
 = 0 if no such dataset exists
 = 1 if no codewords match
 = 2 if the file is being edited by another user
 ≥ 3 if disk errors occur

datain transfers data from disk to the desired buffer

datain *starting record number; to starting location; num records*
 (optional)

dataout transfers data from a buffer to disk

dataout *starting record number; from starting location; num records*
 (optional)

NOTE: With **-datain-** and **-dataout-** the second argument in the tag may be: *student var* OR *c, ECS common location* OR *s, ECS storage location*. The number of records transferred cannot exceed the capacity of the buffer. If omitted, the number of records transferred is 1.

zreturn = -1 if **-datain-** or **-dataout-** is executed successfully
 = 0 if there is a pack error
 = 1 if there is a file error or if there is no preceding successful **-dataset-** command
 = 2 if record numbers extend out of range
 = 3 if the required buffer locations extend out of range
 = 4 (**-dataout-** only) if write permission has not been granted
 = 5 (**-dataout-** only) if record(s) is reserved
 ≥ 6 if there are disk errors

reserve sets system variable "zreturn" in order to allow a user to reserve the common or specified dataset records to prevent changes by more than one user at a time

reserve common

Note: zreturn = -2 if the common is already reserved by this user
 = -1 if -reserve- is executed successfully by this user
 = station number of user who has already reserved the common

reserve dataset, starting record number, num records

Note: zreturn = -2 if these records are reserved by this user
 = -1 if -reserve- is executed successfully by this user
 = $\emptyset$ if there is no successful preceding -dataset-command
 = 1 if record numbers extend out of range
 = 2 if read-only access is allowed
 = 5+station number of user who has reserved these records

release sets system variable "zreturn" to allow the common or indicated dataset records to be released

release common

Note: zreturn = -2 if the common is not reserved by any user
 = -1 if -release- is executed successfully by this user
 = station number of user who has reserved the common

release dataset, starting record number, num records

Note: zreturn = -2 if these records are not reserved by any user
 = -1 if -release- is executed successfully by this user
 = $\emptyset$ if there is no successful preceding -dataset-command
 = 1 if record numbers extend out of range
 = 5+station number of user who has reserved these records

backgnd (no tag) flags a lesson as "background" lesson so that it may use large amounts of CPU time when time is available; when CPU time is scarce, the lesson is handled at lower priority than non-background lessons

foregnd (no tag) switches lesson to foreground processing; normal state of execution

System variables for calculating

lcommon length of common (set by -common- command)

lstorag length of storage (set by -storage- command)

zbpc number of bits per character (currently 6)

zbpw number of bits per word (currently 60)

zcpw number of characters per word (currently 10)

zdsrecs number of records in the dataset in use (=0 if no dataset is in use)

zdswwr number of words per record in the dataset in use (=0 if no dataset is in use)

DATA KEEPING

Requesting data

`dataon` specifies that student data for the lesson is to be collected and sent to a datafile if student records have been set to allow collection of data (see also system variable "dataon")

`dataon (b)` (turns on all data collection)

`dataon tag` (tag can be ok, no, unrec no, vocab, area, output, help, help no, term, term no, errors, signin)

Note: Non-blank tag will temporarily override options set in course records until turned off by `-dataoff-` with appropriate tag.

`dataoff` specifies termination of collection of data for that lesson

`dataoff (b)`

`dataoff tag` (tag is same as used in a preceding `-dataon-`)

Note: Non-blank tag of `-dataoff-` will only turn off options turned on by a previous `-dataon-` with a non-blank tag; `-dataoff-` does not override options set in course records.

Classifying data

area specifies a section in the lesson (called an area) typically representing 5 to 15 minutes of student contact time for which certain data is collected

area *name* (maximum of 10 characters in name; cannot start with a number; variable tag must be enclosed in parentheses)

information collected:

elapsed time in the area (accurate to .1 minute)
 number of arrows encountered
 number of "ok" judgments
 number of responses judged "ok" on 1st attempt
 number of anticipated "no" judgments
 number of unanticipated "no" judgments
 number of term requests satisfied
 number of term requests not satisfied
 number of help-type requests satisfied
 number of help-type requests not satisfied
 whether the area has been completed

area (b) (causes data for the preceding area to be placed in the datafile; no further data is stored until -area- command with nonblank tag is executed)

area incomplete (terminates the current area and flags it as incomplete)

area cancelled (cancels all data for the current area; does not initiate a new area nor produce any records in the datafile)

output puts comment and/or value of an expression into the datafile

output *phrase and/or expr* (formats for the expression are a, n, o, v with embedded form, i.e., *<format,expr>*)

output1 places labeled information from specified student variables into the datafile

output1 *label (optional), starting var, num vars* (maximum of 10 characters in label and 20 consecutive variables)

Transferring data

readset establishes a link between a datafile or course and the lesson which will receive the data

readset *course name*, 'change (or inspect) code on course' (optional),
var for storing num records (optional)

readset *datafile name*, 'change (or inspect) code on datafile' (optional)

Note: The second argument is included if codewords on the lesson and datafile or course do not match.
Variable arguments must be enclosed in parentheses; if the codeword is a variable, quote marks are omitted.

zreturn = -2 if the course exists and is not empty
 = -1 if the datafile exists and is not empty
 = 0 if the name specified is not a datafile or a course
 = 1 if codewords on the lesson and datafile or course do not match
 = 2 if the datafile or course is empty
 = 3 if there is no room in ECS for the buffer
 = 4 if there is a disk error

readd transfers data from a datafile into student variables or central memory variables (must be preceded by **-readset-** naming the datafile)

readd *area, starting var, num vars*

Note: Area summary data consists of the following information:

n(x) OR **pc(x)** = starting variable
n(x) and **n(x+1)** contain the sign-on name (up to 18 characters)
n(x+2) contains lesson name
n(x+3) " area name
n(x+4) " elapsed time for the area (in milliseconds)
n(x+5) " number of arrows for the area
n(x+6) " number of "ok" judgments for the area
n(x+7) " number of "ok" judgments on 1st attempt
n(x+8) " number of anticipated "no" judgments (matched by
 -wrong-, **-wrongc-**, **-wrongu-**, **-wrongv-**, or **-miscon-**)
n(x+9) " number of unanticipated "no" judgments
n(x+10) " number of help-type requests satisfied
n(x+11) " number of help-type requests not satisfied
n(x+12) " number of term requests satisfied
n(x+13) " number of term requests not satisfied
n(x+14) " = -1 if the area was completed, 0 if not

(Continued on next page)

`readd output1,starting var,num vars`

Note: Data consists of the following information:

`n(x)` OR `nc(x)` = starting variable
`n(x)` contains length of `output1` (number of variables)
`n(x+1)` and `n(x+2)` contain the sign-on name (up to 18 characters)
`n(x+3)` contains lesson name
`n(x+4)` " area name
`n(x+5)` " execution time of `-output1-` command in milliseconds
`n(x+6)` " `-output1-` label
`n(x+7)` to `n(x + num vars - 1)` contain data in tag of `-output1-`

`readd signoff,starting var,num vars` (up to 5 variables)

Note: Signoff data consists of the following information:

`n(x)` OR `nc(x)` = starting variable
`n(x)` and `n(x+1)` contain sign-on name (up to 18 characters)
`n(x+2)` contains the lesson name
`n(x+3)` " elapsed time (in minutes) spent in the lesson during this session
`n(x+4)` " total time (in minutes) to complete the lesson if lesson is completed during this session OR -1 if lesson was not completed during this session

Note: With all tags for `-readd-`,

`zreturn` = -1 if there is more data in the datafile
 = 0 if the end of the datafile is reached

`readr` allows access to course roster, to course records of user whose name is in specified variables, or to records of all users in sequence (must be preceded by `-readset-` naming the course)

`readr roster,starting position in roster;to starting var;`
`num names` (reads only names from the roster; each name requires two variables)

`readr name,starting var containing name`
`stats,starting word;to starting var;num vars`
 (reads course statistics)
`rvars,starting address;to starting var;num vars`
 (reads router variables)
`svars,starting address;to starting var;num vars`
 (reads student variables)

`readr sequential`
`stats,starting word;to starting var;num vars`
 (etc., same options as for individual user; if preceded by `-readr name-`, `-readr sequential-` starts reading the roster at the next name; otherwise it starts reading at the beginning of the roster; use `-readr sequential-` in a loop)

Note: With "stats" option:

word 1 and word 2 contain the sign-on name (up to 18 characters)
(display with -showa-)

word 3 contains user type (display with -showa-)

word 4 " date record was created (display with -showa-)

word 5 " date user was last on system (display with -showa-)

word 6 " hour user was last on system (display with -showa-)

word 7 " total time (in hours) on system (display with -show-;
use v-variable)

word 8 " cpu usage in thousand instructions per second
(display with -show-; use v-variable)

word 9 " total number of days on system (display with -show-)

word 10 " number of sessions on system (display with -show-)

zreturn = -1 if there is more data in the course
= 0 if the last record has been read
= 1 if the name is not in the course

Student sign-on and sign-off

restart specifies unit (and lesson if given) where student is to begin on next sign-on

restart (b) (start at main unit containing this command)
 restart unit name (start in this lesson at named unit)
 restart lesson name, unit name (start in named lesson at named unit)
 restart <leslist position>, unit name (start in lesson at specified
 leslist position, in named unit; variable unit name
 must be enclosed in parentheses)

finish sets pointer to the unit which will be executed upon exit from the lesson via STOP1 but not via -end lesson-

finish name
 finish (b) OR finish q (turns off finish setting)
 finish expr, name1, name2, q, name3, x (example of conditional form;
 maximum of 100 arguments in conditional tag)

System variables containing student data information

aarea name of current area (left-justified) (display with -showa-)

aarrows number of arrows encountered

ahelp number of successful help requests

ahelpn number of unsuccessful help requests

aok number of "ok" judgments

aokist number of "ok" judgments on 1st attempt

asno number of specific "no" judgments

aterm number of terms found

atermn number of terms not found

atime elapsed time in area (in milliseconds)

auno number of unanticipated "no" judgments

dataon = -1 if data collection is turned on
 = 0 if data collection is off
 (see also command -dataon-)

NOTE: These system variables are all zeroed at the beginning of an area.

JUDGING

Two types of commands are described in this section: judging commands and regular commands. (In all other sections of this book only regular commands are described.) Regular commands are not executed during the judging process, i.e., after the student has entered a response, nor are judging commands executed before the student has entered a response or in situations where no response is involved. (See The TUTOR Language and lesson "aids" for extensive descriptions of the judging process.)

Regular commands in this section include: -eraseu-, -force-, -edit-, -arrow-, -long-, -jkey-, -copy-, -endarrow-, -iarrow-, -judge-, -okword-, and -noword-. The commands -list-, -endings-, -vocabs-, and -vocab- are special non-executable commands which establish lists of words for use with certain response-matching commands. The -join- command is both regular and judging. The remaining commands in this section are judging commands.

Prepare response

NOTE: All commands in this sub-section are regular commands.

eraseu (regular command) the specified unit is executed at all subsequent arrows in a unit when the student erases part or all of a response after receiving judgment; useful for erasing complicated displays which are not handled by the standard judging process

eraseu *name*

eraseu *(b)* OR **eraseu** *q* (turns off eraseu setting for remainder of unit)

eraseu *expr, name1, name2, q, name2, x* (example of conditional form; maximum of 100 arguments in conditional tag)

force (regular command) alters the input of responses by forcing judging or erasing or by forcing the keypresses into the alternate character set or through the microtable; setting is cleared at every main unit

force *long* (initiates judging when the character input reaches the value of the tag of the -long- command; unnecessary with -long 1-)

force *font* (inserts the font code before the first keypress)

force *micro* (forces all keypresses through the microtable conversion)

force *firsterase* (erases incorrect response and contingent messages when student presses any key, not just NEXT, ERASE, etc.)

force *(b)* (clears previous force settings in that unit)

force *micro, long* (may combine tags)

edit (regular command) required for the EDIT key to be active when the tag of **-long-** exceeds 150; specifies the starting variable of a buffer for storing the characters in a response

edit *starting var* (use student variable)

edit *(b)* (clears edit buffer; if placed after **-arrow-**, prevents use of EDIT key; see also **-inhibit edit-**)

arrow (regular command) plots the response arrow at the specified screen location (see **-inhibit arrow-**); sets defaults: **-long 150-** and **-jkey (b)-**

arrow *coarse*

arrow *finer, finey*

NOTE: To affect the first response, the following commands (**-long-**, **-jkey-**, and **-copy-**) must follow the **-arrow-** command but must precede any judging commands. However, after the student enters a response (e.g., an incorrect response), these commands can be executed among the regular commands following the matched response in order to affect the next response at the same arrow.

long (regular command) sets the maximum number of characters in a response (default is 150 characters); must follow **-arrow-**

long *num characters* (value of tag from 1 to 300; **-long 1-** causes automatic judging; tag > 150 requires use of **-edit-** for EDIT key to be active)

jkey (regular command) specifies function key(s) which will initiate judging (in addition to the NEXT key); must follow **-arrow-**

jkey *keyname* (name of key is in lower case)

jkey *keyname1, keyname2, keyname3, ...*

jkey *(b)* (clears all previous settings with **-jkey-** so that only NEXT key initiates judging)

copy (regular command) specifies the starting variable of the character string which is to be copied on the screen, one word at a time, when the COPY key is pressed; must follow **-arrow-**

copy *starting var, num characters* (use student variable)

endarrow (regular command) (no tag) terminates judging with an unanticipated "no" judgment if the response has not been matched; during the search for additional **-arrow-** commands after an "ok" judgment, terminates **-arrow-** search and switches back to the pre-arrow state

Establish lists

list (non-executable) sets up list of synonyms for judging; used with -answer-, -wrong-, -answerc-, -wrongc-, and -match-

list *listname, word1, phrase*consisting*of*several*words, word2, word3, ...* (maximum of 7 characters in listname)

endings (non-executable) used with -vocabs- and -vocab- to add endings to root words (must precede -vocabs- and -vocab-)

endings *number, ending1, ending2, ...* (number is an integer from 0 to 9)

Note: In -vocabs- or -vocab-,

word/number adds endings to the root word and includes all words in the vocabulary

word//number adds only words with endings to the vocabulary; the root word is not included

Up to 10 different -endings- commands with up to 7 different endings each may be included; apostrophe is legal in an ending.

vocabs (non-executable) sets up lists of ignorable words and synonymous important words; used with -concept- and -miscon-; does capitalization and spelling checks

vocabs *name*

<ignorable words separated by commas>

*word1, word2, phrase*consisting*of*several*words*

(synonymous words3 and phrases separated by commas)

(word4/s, word5/ending1/ending2, word6//ending1)

word7/number1, word8//number2

(word1=1, word2=2, synonym2=2, ...)

vocab (non-executable) similar to -vocabs- except does not check for capitalization and spelling and does not allow phrases

vocab *name*

<ignorable words separated by commas>

word1, word2

(synonymous words3 separated by commas)

(word4/s, word5/ending1/ending2, word6//ending1)

word7/number1, word8//number2

NOTE: Up to 7 characters are permitted in the name of the vocabulary. Up to 7 different endings may be added to one root. When sets of endings are used repeatedly, -endings- plus -vocab(s)- is more convenient than -vocab(s)- with actual endings included.

Modify judging copy of response

bump removes specified characters from the judging copy of the response before judging

bump *characters* (maximum of 8 characters; use additional -bump- commands for more than 8 characters)

put replaces a character string in the judging copy of the response with another character string

put *string1=string2* (replace *string1* with *string2*)

putd similar to -put- but uses the first character in the tag as the separator between strings

putd */string1/string2/* (separator is /)

putd *,string1,string2,* (separator is ,)

putv similar to -put- but works with stored strings

putv *starting var of string (left-justified), num characters in string, starting var of replacement string (left-justified), num characters in replacement string*

NOTE: Maximum number of characters in strings for -put-, -putd-, and -putv- is 50. If the replacement operations cause the judging copy of the response to exceed 300 characters, judging terminates with a "no" judgment.

close takes characters stored in the right-most six bits from successive variables and makes a judging copy for use with judging commands (often paired with -open-); the end of the character string is indicated by the specified number of characters or by the first occurrence of six bits equal to zero, whichever is attained first

close *starting var, num characters* (use n-variables)

loada takes the characters stored in the specified variable(s) by -storea-, -pack-, or -calc- and makes a judging copy; the end of the character string is indicated by the specified number of characters or by the first occurrence of six bits equal to zero, whichever is attained first

loada *starting var, num characters (optional)* (number of characters, if omitted, is 10)

Modify judging procedure

NOTE: The various -specs- options do not affect all judging commands. With each -specs- option are the numbers of the commands affected by that option from the following list.

judging commands affected by -specs-

1. -match-
2. -answer-, -wrong-, -answer-, -wrong-
3. -concept-, -vocab-, -miscon-
4. -concept-, -vocab-, -miscon-
5. -exact-, -exact-, -exact-
6. -ansv-, -wrongv-, -ansu-, -wrongu-, -store-, -storeu-
7. -store-
8. -storea-

specs allows control over processing of responses; also serves as a marker for execution of subsequent regular commands after judging is complete

specs	allwords	(treats integers like letters (rather than numbers) so that a number-letter boundary is not like a word-word boundary or punctuation)
		(with 1, 2, 3, 4 above)
specs	alphxnum	(treats a letter-number boundary like a word-word boundary or like punctuation)
		(with 1, 2, 3, 4, 7 above)
specs	bumpshift	(removes shift codes from the judging copy of the response)
		(with all commands above)
specs	exorder	(checks the order of ignorable words)
		(with 2 above)
specs	nodiff	(turns off the numeric difference judger, which treats a numerical response as a "misspelling" if it is within 10% of the correct response; no spelling markup is done)
		(with 2 above)
specs	nomark	(turns off answer markup)
		(with all commands above where markup is done: 2, 3, 4)
specs	nookno	(prevents appearance of "ok" and "no")
		(with all commands above)
specs	noops	(prevents use of mathematical operations in a numerical response)
		(with 6 above)

specs noorder (turns off the order judger; allows any word order)
(with 2, 3, 4 above)

specs nospell (turns off the spelling judger; no spelling markup is done)
(with 2 above)

specs novars (prevents use of variables defined in define set "student")
(with 6 above)

specs okassign (allows assignment of a value to a variable defined in define set "student")
(with 6 above)

specs okcap (allows a capitalized word in the response to match a non-capitalized word in tag of response-matching command)
(with 1, 2, 3 above)

specs okextra (allows extra words in the response; caution--words not in -vocabs- list may be treated as spelling errors)
(with 2, 3, 4 above)

specs okspell (allows any reasonable spelling)
(with 1, 2, 3 above)

specs toler (allows 1% tolerance in a numerical response)
(with 1, 2 above)

specs (b) (acts only as a marker)

specs nookno,okspell,noorder (may combine tags)

Note: The following system variables are set properly even if use of the -specs- tag causes the response to match the tag of a response-matching command.

-specs- tag	system variable
okspell	spell
okcap	capital
okextra	extra
noorder	order

Store judging copy of response

NOTE: Commands -store-, -storeu-, and -storen- terminate judging if an error is found in the form of the response.

store calculates the numerical value of response and stores it in the variable specified in the tag

store *var*

storeu similar to -store- but also evaluates dimensionality of units

storeu *var* where number is stored, starting *var* where powers of dimensions are stored (see -define-, units)

storen searches for and evaluates simple numerical expressions (without variables) in the response, which may also contain non-numeric characters, and stores numerical results in the specified variables one at a time; removes numerical parts of response from the answer buffer and replaces them with blanks; numerical parts must be set off from words by spaces or other punctuation; if no numerical expression is found, the variables are set to 0 and judging ends with a "no" judgment

storen *var1*

storen *var2*

:

storea stores characters from the response, left-justified, in the specified variable(s), 10 characters per variable; unused character positions are filled with octal zeros

storea *starting var, num characters (optional)* (number of characters, if omitted, is 10)

Note: Use n-variable(s) for storing character string if subsequent comparison for equality with another string is done. (If character string is stored for other purposes, v-variables are acceptable.) Segmented variables cannot be used with -storea-.

open places the characters in the response, one-by-one, in the right-most six-bits of successive variables starting at the specified variable

open *starting var* (use n-variables)

Match judging copy of response

NOTE: In this sub-section references to response mean judging copy of response.

NOTE: Up to 40 "words" (entries separated by space or punctuation) are permitted in responses with -match-, -answer-, -wrong-, -answer-, -wrong-, -concept-, and -miscon-. If the number of words exceeds 40, judgment is "no" and "anscnt" is set to -2.

match checks the response against a list and sets a variable to the relative position of the matched character string; if no match is found, the variable is set to -1 and judgment is "no"; otherwise judgment is "ok"; always ends judging

for example:

```
match var,word0,word1,phrase*words2,word3
match var,(word0,synonym0),(word1,(listname1)),((listname2))
```

answer compares the response with the tag; checks for word order, spelling, capitalization, extra words, and numeric tolerance unless altered by -specs-

for example:

```
answer words and phrase*words (blank tag matches a response in
which nothing is entered or which contains only spaces
and punctuation)
answer <extra words>(synonymous words1 and phrase*words1 separated
by commas)(synonymous words2 and phrase*words2 separated
by commas)word3
answer <<listname1>> ((listname2),word2)
answer response1;response2;response3 (each argument may have
any of the preceding forms of the tag of -answer-;
alternate responses for the same argument are separated
by commas)
```

Note: Tag must not contain punctuation marks, although the student's response may.

wrong compares the response with the tag; performs checks available with -answer-

wrong words and phrase*words

Note: Options for the tag of -wrong- are the same as for the tag of -answer- but for an incorrect student response.

answerc compares the response with one of several arguments in the tag, depending on the value of a conditional expression; performs the checks available with **-answer-**

answerc *expr;responsem;response0;;response2* (the arguments may have any of the forms allowed in the tag of **-answer-**; a blank argument indicates no anticipated response for that value of the conditional expression)

wrongc same options as for **-answerc-** but for an incorrect response

wrongc *expr;responsem;;;response2;response3*

concept compares the response with the tag; **-vocab-** or **-vocabs-** provides synonyms

concept *words and phrase words* (no asterisk in phrases; blank tag matches a response which does not contain any keywords from the vocabulary)

concept *word1 word2,var1 $\Leftarrow$ word1,var2 $\Leftarrow$ word2* (detects which synonym is entered if the vocabulary is appropriately set up)

miscon same options as for **-concept-** but for an incorrect response

miscon *words and phrase words*

exact compares the response with the tag for an exact character string match

exact *string* (blank tag matches a response in which nothing is entered)

exactc checks the response for an exact character string match with one of several arguments in the tag, depending on the value of a conditional expression

exactc *expr,stringm,string0,,string2*

exactv compares the response with a stored character string for an exact match; the end of the character string is indicated by the specified number of characters or by the first occurrence of six bits equal to zero (000000), whichever is attained first

exactv *starting var of stored string, num characters (optional)*
 (number of characters, if omitted, is 10; if the number of characters is 0, the response is judged correct if nothing is entered)

NOTE: With the following four commands (*-ansv-*, *-wrongv-*, *-ansu-*, *-wrongu-*), tolerance is optional. When tolerance is omitted, the default is 10^{-9} if the absolute value of the tag value is less than approximately 100 OR $(10^{-11} \times |\text{tag value}|)$ if the absolute value of the tag value is greater than approximately 100. These commands cannot judge values smaller in absolute value than 10^{-9} since any response less than 10^{-9} will then match the tag.

ansv checks a numerical response against the first argument in the tag, with tolerance, if given, set by the second argument

ansv *correct value, tolerance (absolute deviation or % deviation)*

wrongv similar to *-ansv-* but for the incorrect numerical response

wrongv *incorrect value, tolerance (absolute deviation or % deviation)*

ansu similar to *-ansv-* but checks for correct units

ansu *number and units, tolerance (absolute deviation or % deviation)*

wrongu similar to *-ansu-* but for incorrect response

wrongu *number and units, tolerance (absolute deviation or % deviation)*

wrongu *number, tolerance (absolute deviation or % deviation)*

(may be used to indicate that units are missing)

NOTE: For applications of *-ansu-* and *-wrongu-* see *-storeu-* and *-define-*. Commands *-ansv-* and *-wrongv-* accept defined units in the student's response but do not compare with author's units.

touch specifies the location of the touch square for a touch response
(see -enable- and -disable-)

touch *center coarse, tolerance in touch squares (optional)*
(tolerance, if omitted, is 1; blank tag matches any touch input)

touch *lower left coarse, width in touch squares, height in touch squares*

touch *location1, tolerance1; location2, tolerance2; ...* (up to 20
one-, two-, or three-argument touch locations may be specified)

Note: One touch square is 32 dots on each side. Numbering of touch squares on the screen is: 0,0 at lower left; 0,15 at upper left; 15,0 at lower right; 15,15 at upper right.

touchw same options as -touch- except gives a "wrong" judgment

touchw *center coarse, tolerance (optional)*

touchw *lower left coarse, width in touch squares, height in touch squares*

or (no tag) placed on the line between response-matching commands to provide alternative responses for the same value of "anscnt"

ans (no tag) allows use of the ANS key; terminates judging only if ANS is pressed; otherwise normal judging occurs; -ans- must be the first judging command following the -arrow- command

Specify judgment unconditionally

ok (no tag) judges any response "ok"

no (no tag) judges any response "no"

ignore (no tag) erases and ignores any response; stops further processing
and waits for a new response

Reference other units which may contain judging commands

join causes execution of the specified unit without change of main unit; commands following **-join-** are executed; **-join-** is executed in all states: regular, judging, and search (also see description under SEQUENCING, Automatic sequencing)

join *name*

join *expr, name1, name2, x, name2* (example of conditional form)

iarrow (regular command) the specified unit is executed after each subsequent **-arrow-** in a unit before the first judging command is executed

iarrow *name*

iarrow *(b)* OR **iarrow** *q* (turns off **iarrow** setting for subsequent arrows in the unit)

iarrow *expr, name1, name2, q, name2, x* (example of conditional form; maximum of 100 arguments in conditional tag)

Note: The **-iarrow-** command is equivalent to a **-join-** command after each subsequent **-arrow-** (before the first judging command); the specified unit is executed in all states.

Alter judgment

judge (regular command) alters the judgment rendered by judging commands

judge ok (sets judgment to "ok"; continues processing)

judge no (sets judgment to "no" (unanticipated); continues processing)

judge wrong (sets judgment to "no" (anticipated); continues processing)

judge okquit (sets judgment to "ok"; terminates processing at that arrow except for regular commands following -specs-)

judge noquit (sets judgment to "no"; terminates processing at that arrow except for regular commands following -specs-)

judge quit (leaves judgment unchanged and terminates processing at that arrow except for regular commands following -specs-; allows the student to proceed to the next arrow even if judgment on the current arrow is "no")

judge ignore (stops all processing, erases the response, and waits for a new response)

judge exit (rescinds judgment and waits for additional keys)

judge continue (resumes judging using the modified response, as altered by -bump-, -put-, -specs-, -match-, -storen-, etc.)

judge rejudge (resumes judging using the original, unmodified response and clears the -specs- setting)

judge *expr*,x,no,ignore,ok (example of conditional form; argument x leaves judgment unchanged)

Alter feedback

okword (regular command) permits "ok" message to be changed

okword *new word for use with "ok" judgment* (may be blank)

noword (regular command) permits "no" message to be changed

noword *new word for use with "no" judgment* (may be blank)

NOTE: Tags of -okword- and -noword- may have up to 9 characters.
A space is automatically provided before the message.
Commands -okword- and/or -noword- may be placed anywhere in the lesson. Once they are executed, they are in effect until execution of another -okword- and/or -noword- command.

System variables for judging

judging in general

anscnt number of response-matching commands encountered at an arrow
 before finding one that matches student response; otherwise,
 = -2 if student response contains more than 40 words
 = -1 if no tag is matched
 = 0 for a store error
 zeroed for each -arrow- and each -specs- command

ansok = -1 if the response is a satisfactory match to the preceding
 response-matching command
 = 0 otherwise;
 in particular, after -no-,
 = -1 if there is no match at all
 = 0 if match is poor

jcount number of internal 6-bit character codes in response

judged = -1 for any "ok" judgment
 = 0 for any "wrong" judgment (anticipated "no")
 = +1 for any "no" judgment (unanticipated "no")

key [See description under system variables for sequencing.]

ntries number of attempts on the current arrow

verbal responses

judging commands which affect system variables for verbal responses:

1. -match-
2. -answer-, -wrong-, -answer-, -wrong-
3. -concept-, -vocabs-, -miscon-
4. -concept-, -vocab-, -miscon-

capital = -1 if there are no capitalization errors, = 0 otherwise
 (with 1, 2, 3 above)

entire = -1 if all important words are present, = 0 otherwise
 (with 2 above)

extra = -1 if there are no extra words in response, = $\emptyset$ otherwise
(with 2, 3, 4 above)

order = -1 if word order is ok, = $\emptyset$ otherwise
(with 2 above)

phrase = -1 if there is no phrase error, = $\emptyset$ otherwise
(with 1, 2, 3 above)

spell = -1 if spelling is ok, = $\emptyset$ otherwise
(with 1, 2, 3 above)

wcount number of words in response
(with all above)

numerical responses

These system variables are set with -ansv-, -wrongv-, -ansu-, -wrongu-, -store-, -storeu-, and -compute- (which is not a judging command).

formok gives diagnostics on errors in mathematical expressions

- = -1 if the expression is ok
- = $\emptyset$ if there is a bad function argument or index
- = 1 if there is an illegal character
- = 2 if there are unbalanced parentheses
- = 3 if there are too many decimal points
- = 4 if there are variables not defined in define set "student"
- = 5 if a symbol involving \$ is not a logical operator
- = 6 if the expression has bad form
- = 7 if a value is assigned to a non-variable
- = 8 if an octal constant contains digit 8 or 9
- = 9 if there is an error in an alpha string
- = 10 if a number has too many digits
- = 11 if an array index is out of bounds
- = 12 if there are variables with -specs novars-
- = 13 if there are operations with -specs noops-
- = 14 if there are assignments without -specs okassign-
- = 15 if units in the response are used incorrectly
- = 16 if too much computing is attempted

opcnt number of arithmetic operations and functions in the response

varcnt number of defined variables and functions (define set "student")
in the response

PRESENTING

Text

at specifies starting position of display on plasma panel

at coarse
at finex, finey

Note: The following formulas convert between character grid and fine grid.

finex = $800 \text{frac}(\text{coarse}/100) - 8$
finey = $512 - 16 \text{int}(\text{coarse}/100)$
coarse = $100 \{1 + \text{int}[(511 - \text{finey})/16]\} + \text{int}(\text{finex}/8) + 1$

atnm similar to -at- but does not reset the left margin

atnm coarse
atnm finex, finey

write displays text on plasma panel

write Any message, which may consist of several lines and include embedded information.

writec displays one of several messages depending on the value of an expression

writec *expr* *message0* *message1* *message2* ...
(with general delimiter)

NOTE: The embed feature may be used with -write- and -writec-. See descriptions of the individual commands in this section for information on format and minimum, which are optional.

embedded -show- $\langle \text{show}, \text{expr}, \text{format}, \text{minimum} \rangle$ or $\langle \text{s}, \text{expr}, \text{format}, \text{minimum} \rangle$
embedded -showz- $\langle \text{showz}, \text{expr}, \text{format} \rangle$ or $\langle \text{z}, \text{expr}, \text{format} \rangle$
embedded -showt- $\langle \text{showt}, \text{expr}, \text{format} \rangle$ or $\langle \text{t}, \text{expr}, \text{format} \rangle$
embedded -showe- $\langle \text{showe}, \text{expr}, \text{format} \rangle$ or $\langle \text{e}, \text{expr}, \text{format} \rangle$
embedded -showo- $\langle \text{showo}, \text{expr}, \text{format} \rangle$ or $\langle \text{o}, \text{expr}, \text{format} \rangle$
embedded -showa- $\langle \text{showa}, \text{expr}, \text{format} \rangle$ or $\langle \text{a}, \text{expr}, \text{format} \rangle$
embedded -at- $\langle \text{at}, \text{coarse} \rangle$
 $\langle \text{at}, \text{finex}, \text{finey} \rangle$
embedded -size- $\langle \text{size}, \text{expr giving size of writing} \rangle$
embedded -rotate- $\langle \text{rotate}, \text{expr giving angle for writing} \rangle$
 $\langle \text{m}, \text{w} \rangle$ (write mode)
embedded -mode- $\langle \text{m}, \text{e} \rangle$ (erase mode)
 $\langle \text{m}, \text{r} \rangle$ (rewrite mode)

NOTE: In the following six commands (-show-, -showz-, -showt-, -showe-, -showo-, -showa-), the general form is

`showz expr,format` (if format is $\emptyset$, nothing is displayed)

where the format, which may also be an expression, is optional.

For displaying arrays the general form is

`showt arrayname,format` (maximum of 16 rows and columns)

`show` displays the value of a variable or an expression with the specified number of significant digits but with suppression of trailing zeros after decimal point; exponential format is displayed if number of digits preceding decimal point exceeds specified format by more than 4, or if the value is less than 10^{-4} ; minimum is between $\emptyset$ and 1 and specifies smallest non-zero value to be displayed ($\emptyset$ is displayed if absolute value of expression is less than minimum)

`show expr,num digits,minimum` (format, if omitted, is 4; minimum, if omitted, is 10^{-9})

`showz` similar to -show-, but all digits, including trailing zeros, are displayed

`showz expr,num digits` (format, if omitted, is 4)

`showt` displays the value of a variable or an expression in the specified format

`showt expr,num digits preceding decimal point,num digits following decimal point` (may be omitted if zero)
(format, if omitted, is 4, 3 for v-variable, 8 for r-variable; if the number of decimal places is less than 10, format may also be expressed as a single decimal number: e.g., 4.3 is equivalent to 4,3)

`showe` displays the value of a variable or an expression in exponential format with the specified number of significant digits, including a leading blank or a minus sign

`showe expr,num digits` (format, if omitted, is 4)

`showo` displays the value of a variable or an expression in octal notation (base 8)

`showo expr,num digits displayed` (format, if omitted, is 21; in embedded -showo- format is 20)

showa displays characters from variable(s) or string specified in tag

showa *'string'* (string must contain ≤ 10 characters)
 showa *starting var, num characters* (format, if omitted, is 10)

erase erases screen, selectively or entirely

erase *(b)* OR erase *negative value* (causes full-screen erase)
 erase *num characters to be erased*
 erase *num characters per line, num lines* (causes block erase)

mode specifies terminal writing mode (see also system variable "mode")

mode write (normal writing state; writes only specified dots)
 mode erase (erases only specified dots)
 mode rewrite (erases and rewrites in one step; does not work
 with "size" $\neq 0$)
 mode *expr, erase, write, x, erase* (example of conditional form;
 argument x leaves writing mode unchanged)

Note: Mode is reset to "write" after any full-screen erase, in particular at a main unit not containing -inhibit erase-.

color displays screen information in specified color

color orange (same effect as -mode write-)
 color black (same effect as -mode erase-)

size tag specifies size of subsequent line-drawn characters; remains in effect across main unit boundaries until turned off explicitly

size *expr giving size of characters*
 size 0 (restores to standard writing)

rotate causes line-drawn characters to be written at angle specified in tag; remains in effect across main unit boundaries until turned off explicitly (must be used with -size- tag $\neq 0$)

rotate *expr giving angle in degrees (omit degree symbol),*
 measured counter-clockwise from horizontal
 rotate 0 (restores to horizontal writing)

inhibit temporarily disables certain normal actions of TUTOR in a unit; cleared at each main unit

inhibit anserase	(prevents automatic erasure of answer-contingent message when response is erased)
inhibit arrow	(prevents plotting of arrow on screen)
inhibit blanks	(prevent judging if NEXT is pressed before any characters are typed)
inhibit charclear	(prevents clearing of charset flag)
inhibit dropstor	(prevents storage from being dropped during jumpout)
inhibit edit	(prevents use of EDIT key)
inhibit erase	(prevents normal full-screen erase when proceeding to next main unit)
inhibit from	(prevents return to the lesson containing this statement via -jumpout return- or -jumpout return,return-)
inhibit jumpchk	(prevents ECS check before attempting a jumpout)
inhibit term	(prevents use of TERM key)
inhibit unload	(prevents unloading of common or storage from CM to ECS)
inhibit (b)	(removes effect of previous -inhibit-s in that main unit)
inhibit arrow,blanks,erase	(may combine tags)

delay permits short delays during output

delay *duration of output delay in fractions of a second, up to 1 second (accuracy to 1/60 second)*

Note: This command causes "do-nothing" output to be sent to the terminal for specified delay time.

char permits specification of specially designed characters for display

char *name, arg1, arg2, ..., arg8*

Note: Name may be a number from 0 to 126 (excluding 63) or a defined name. Arguments arg1, arg2, ..., arg8 are numbers which specify which of the 16 dots are lit in each of the 8 columns of the character space.

for example:

```
define chi=88      $$ load chi on X
char  chi,o4020,o10040,o6300,o1600,o3140,o4020,o10040,0
```

plot displays special characters previously specified by -char- commands

plot *name*

Note: Special characters may also be displayed by pressing FONT and then the key(s) where the character(s) are loaded in the terminal memory. Built-in characters are displayed after FONT is pressed again.

dot coarse
dot finex, finey

for example:

```

draw      coarse      (draws dot; -dot- command is faster if
draw      finex, finey many dots are to be plotted; -draw- is
                        faster if lines are also being drawn)
draw      coarse1; coarse2
draw      finex1, finey1; finex2, finey2      (draws line)
draw      coarse1; coarse2; coarse3; etc.      (draws
draw      finex1, finey1; finex2, finey2; finex3, finey3; etc. connected
                                                lines)
draw      ;coarse
draw      ;finex, finey      (draws continued line)
draw      finex1, finey1; coarse2; skip; coarse3; finex4, finey4
                        ("skip" moves to new position without plotting)

```

Note: Coarse grid and fine grid coordinates may be mixed.
Maximum number of arguments is 63.

```

rdraw    similar to -draw- but figure is affected by preceding -size- and
         -rotate- or by -rotate- alone; center of rotation is specified
         by last screen location; "wherex" and "wherey" are set to this
         location after execution

```

rdraw tag like **-draw-** except with respect to center of rotation, i.e., in screen dots from center of rotation

circle draws circle with parameters specified in tag

```
circle radius,center finex,center finey,start angle,end angle
      (last two arguments are optional: if omitted, entire
      circle is drawn; radius is in dots; angles are measured
      in degrees counter-clockwise from the horizontal; degree
      sign is omitted)
```

`circleb` draws broken circle

`circleb radius,center finex,center finey,start angle,end angle`
 (last two arguments are optional: if omitted, entire circle is drawn; radius is in dots; angles are measured in degrees counter-clockwise from the horizontal; degree sign is omitted)

`window` delimits area on screen for line-drawn display; remains in effect across main unit boundaries until turned off explicitly

`window lower left coarse,upper right,coarse`

`window minimum finex,minimum finey,maximum finex,maximum finey`

`window (b)` (removes effect of previous `-window-`)

Drawing graphs

origin specifies location of origin; all other plotting and writing with graphing commands are relative to this origin

origin *coarse*

origin *finex, finey*

Note: Upon entering a lesson the origin is automatically set to -origin 0,0-.

NOTE: In the next three commands (-axes-, -bounds-, -frame-) x and y values are in dots relative to the origin.

axes specifies and draws axes (see NOTE above)

axes *negative x, negative y, positive x, positive y*

axes *positive x, positive y* (draws only positive axes)

axes *(b)* (draws last set of axes specified)

Note: To draw one-quadrant axes (other than both positive axes) with labeling on the outside of the axes, use the four-argument form of the tag with arguments corresponding to missing axes set to 0.

bounds establishes boundaries on the axes but does not draw the axes (see NOTE above)

bounds *negative x, negative y, positive x, positive y*

bounds *positive x, positive y*

Note: Upon entering a lesson the boundaries are automatically set to -bounds 511,511-.

frame draws frame around boundaries specified in tag or by previous -axes- or -bounds- command (see NOTE above)

frame *minimum x, minimum y, maximum x, maximum y*

frame *(b)* (draws frame specified by last -axes- or -bounds-)

NOTE: If any of the next four commands (-scalex-, -scaley-, -lscalex-, -lscaley-) are omitted, a linear scale with length set by the preceding -axes- is assumed.

scalex specifies maximum value and value at origin on x axis; remains in effect across main unit boundaries until reset

scalex *maximum value of x, value of x at origin (optional)*
(value at origin, if omitted, is zero)

`scaley` specifies maximum value and value at origin on y axis; remains in effect across main unit boundaries until reset

`scaley` *maximum value of y, value of y at origin (optional)*
(value at origin, if omitted, is zero)

`lscalex` specifies maximum value and value at origin on x axis; scale between these points is proportional to the logarithm of maximum x divided by value at origin; remains in effect across main unit boundaries until reset

`lscalex` *maximum value of x, value of x at origin (optional)*
(value at origin, if omitted, is 1, i.e., 10^0)

`lscaley` specifies maximum value and value at origin on y axis; scale between these points is proportional to the logarithm of maximum y divided by value at origin; remains in effect across main unit boundaries until reset

`lscaley` *maximum value of y, value of y at origin (optional)*
(value at origin, if omitted, is 1, i.e., 10^0)

NOTE: Subsequent graphing commands are in appropriate scaled units.

NOTE: For the following labeling commands,
`marksize` = 0 or omitted for normal label marks
`marksize` = 1 for major marks extending to bounds of graph
`marksize` = 2 for all marks extending to bounds of graph
 Minor mark interval may be omitted. If major mark interval is omitted or set to zero, computer chooses "best" interval.

For labeling log scales; if
`num minor marks` < 0, minor marks are not plotted
`num minor marks` = 0 or 3 (or omitted), minor marks are placed at values of 1, 2, 5 within the decade
`num minor marks` = 5, minor marks are placed at 1, 2, 3, 5, 7
`num minor marks` = 10, minor marks are placed at 1, 2, 3, 4, 5, 6, 7, 8, 9

`labelx` draws tick marks and labels x axis (see NOTE above)

`labelx` *major mark interval, minor mark interval, marksize*
`labelx` *format, num minor marks, marksize* (for labeling log scales; `format` = 0 for the moment)

labeledy draws tick marks and labels y axis (see NOTE above)

labeledy *major mark interval, minor mark interval, marksize*

labeledy *format, num minor marks, marksize* (for labeling log
scales; *format = 0* for the moment)

markx draws tick marks on x axis with no labels (see NOTE above)

markx *major mark interval, minor mark interval, marksize*

markx *format, num minor marks, marksize* (for labeling log
scales; *format = 0* for the moment)

marky draws tick marks on y axis with no labels (see NOTE above)

marky *major mark interval, minor mark interval, marksize*

marky *format, num minor marks, marksize* (for labeling log
scales; *format = 0* for the moment)

locate similar to *-at-* but relative to the origin and in scaled units

locate *x-location, y-location*

graph places dot or other specified character string centered at position indicated relative to origin and in scaled units

graph *x-location, y-location, string (optional)* (maximum of 9
characters in string; if string is not specified, dot
is plotted)

graph *x-location, y-location; var, num characters (optional)*
(if number of characters is omitted, 10 characters are
plotted)

gdraw like *-draw-* but relative to origin and in scaled units

for example:

gdraw *x1, y1; x2, y2* (draws line on graph)

polar changes interpretation of tags of graphing commands following it to polar arguments containing scaled radius and polar angle; also may set scales on x and y axes; remains in effect across main unit boundaries until turned off explicitly; polar conversion and scaling must be turned off independently

polar (b) (turns on polar conversion)
polar *maximum value of x and y* (turns on conversion and scales both axes)
polar *maximum value of x, maximum value of y* (turns on conversion and scales axes independently)
polar *negative value* (turns off polar conversion but not scale, if used)

vector draws vector symbol with specified tail and head locations and head size

vector *xtail,ytail,xhead,yhead,size* (optional)
vector *xhead,yhead,size* (optional) (tail at origin)
vector *distancetail,angletail,distancehead,anglehead,size* (optional)
 (use with -polar-; degree sign is necessary with angle)
vector *length,angle,size* (optional) (use with -polar- for vector with tail at origin; degree sign is necessary with angle)

Note: Size, if omitted, is 10.5 (in dots) for moderate-length arrows. Negative size indicates open arrowhead. Because of the default conditions of -origin 0,0- and -bounds 511,511-, -vector- may be used without preceding -origin- and -bounds-. Tag of -vector- is then in fine-grid screen coordinates.

vbar draws vertical bar relative to origin, in scaled units, centered on x-location

vbar *x-location,height,string* (optional) (maximum of 9 characters in string; if string is omitted, rectangle is drawn)
vbar *x-location,height;var,num characters* (optional)
 (if number of characters is omitted, 10 characters are plotted)

hbar draws horizontal bar relative to origin, in scaled units, centered on y-location

hbar *length,y-location,string* (optional) (maximum of 9 characters in string; if string is omitted, rectangle is drawn)
hbar *length,y-location;var,num characters* (optional) (if number of characters is omitted, 10 characters are plotted)

`delta` specifies stepsize for a following `-funct-` command

`delta` *stepsize*

Note: If `-delta-` is omitted, stepsize is set to 1.

`funct` plots curve specified in tag, with stepsize given by tag of preceding `-delta-`, or plots curve with stepsize specified in tag of `-funct-`

`funct` *function expr, independent var*

Note: Range of independent variable is set by boundaries of `-axes-` and `-scalex-` commands.

`funct` *function expr, independent var* *initial value, final value, stepsize*

Note: If initial or final values of the independent variable are beyond previously set boundaries, the latter are used.
For polar functions default range is 0 to 2π .
Stepsize $\geq .02(\text{final value} - \text{initial value})$

Note: With either form of `-funct-`, a v-variable is recommended for the independent variable.

Non-screen

slide operates slide projector and selects slide specified in tag

slide *slide number* (value from 0 to 255)
 slide *row + 16column* (row, column from 0 to 15)
 slide 512 (turns off lamp)
 slide 256 (closes shutter)
 slide 512 + *slide number* (selects slide with lamp off)
 slide 256 + *slide number* (selects slide with shutter closed)
 slide noslide (selects slide 0, turns off lamp, and closes shutter)

audio plays audio message specified in tag

audio *message number* (value of tag from 0 to 4095)

play plays audio device recording, with message location specified

play *track, sector, num sectors* (128 tracks, 32 sectors each)

record records message at location specified

record *track, sector, num sectors*

enable (no tag) allows inputs from external device other than keyset (e.g., touch panel); must be reset for each arrow in a unit; -enable- in unit with no arrows allows touch anywhere on screen to behave like pressing NEXT

disable (no tag) refuses inputs from external device except keyset (e.g., touch panel); this is the normal state and is reset for each new arrow or full-screen erase

ext sends octal value of tag to external accessory (or to accessory at another station if ext option has been turned on by receiving user)

ext *expr*

ext *expr, station* ("zreturn" is set to -1 if sent, to 0 if not)

Special display

tabset sets tabs which can be used by student pressing TAB key

tabset *octal number containing 10 packed tab settings from left to right* (each setting is a 6-bit octal number; unused settings to the right must be filled with octal zeros)

for example, to set tabs at horizontal character positions 8, 21, 30, 48, 56, and 63, use:

tabset 010 25 36 60 70 77 00 00 00 00

micro specifies micro table definitions used when student presses MICRO key and then another key

micro *lesson name, block name* (enclose variable arguments in parentheses)

micro microtable, standard (specifies built-in micro table definitions)

Note: **zreturn** = -1 if the micro table is available
 = 0 if the micro table is not found

charset causes character set name in tag to be loaded into terminal memory (see -inhibit charclear-)

charset *lesson name, block(s) name* (up to 79 characters in one-block charset, up to 126 characters in two-block charset; enclose variable arguments in parentheses)

charset (b) (clears charset flag)

Note: **zreturn** = -1 if the character set is loaded successfully
 = 0 if the character set is not found
 = 1 if the STOP key is pressed during loading
 = 2 if there is an error in loading
 = 3 if the character set is empty
 = 4 if the variable containing the lesson name is 0
 = 5 if the variable containing the charset name is 0

chartst allows check for presence of character set in terminal memory; sets "zreturn" to -1 if charset flag is set and to 0 if not

chartst *lesson name, block(s) name*

lineset allows design of line-drawn characters which are affected by preceding -size- and -rotate-; "size" must not be $\emptyset$; like -charset- characters, linechars are accessed by the FONT key

lineset *lesson name, block(s) name* (from 1 to 3 blocks; up to 128 small linechars in 1 block; enclose variable arguments in parentheses)

Note: zreturn = -1 if the lineset is attached successfully
 = $\emptyset$ if the lineset is not found
 = +1 if there is an error in the lineset

altfont changes font mode of terminal

altfont on OR altfont 1 OR altfont alt (switches terminal to alternate font)

altfont off OR altfont $\emptyset$ OR altfont normal (switches terminal to normal font, which is the default state)

Note: Tag can be calculated, but it must be exactly $\emptyset$ or 1.
 -altfont- setting remains in effect until reset by another -altfont- or until -jumpout- is executed.

System variables for presenting

mode = -1 for -mode erase- or -color black- see also commands
 = 0 for -mode rewrite- -mode- and -color-
 = +1 for -mode write- or -color orange-

size current value of tag of -size- command (see also command -size-)

where current character-grid location for next display

wherex current fine-grid x location for next display

wherey current fine-grid y location for next display

ROUTING

Router lesson

route used in a router lesson to specify to which unit in the router lesson the student is sent upon leaving the instructional lesson

route end lesson, *name* (exit via -end lesson- or -jumpout q-)
 route error, *name* (exit via execution error)
 route finish, *name* (exit via STOP1)
 route resignon (upon STOP1 exit from a lesson, provides the student with a choice page offering an option to continue working [i.e., return to the router] or to sign off completely)

routvar (non-executable) sets up special variables in a router lesson which can be used only in the router lesson; they are referenced by vr or nr

routvar *num vars* (maximum of 50 variables)

allow used in a router lesson to specify that router common and/or router variables may be referenced in the instructional lesson

allow read (read-only access to router common)
 allow write (read and write access to router common)
 allow read rvars (read-only access to router variables: vr, nr)
 allow (b) (clears last setting of -allow-)

System router

NOTE: The following commands may be used in any context but are particularly useful with the system router.

lesson sets the system variable "ldone" to indicate whether a lesson is considered complete by the system router

lesson completed (sets "ldone" to -1)

lesson incomplete (sets "ldone" to 0)

lessor noend (sets "ldone" to +1; may be used in lessons with no logical end)

lesson *expr*, completed, incomplete, x, noend (example of conditional form; argument x leaves "ldone" unchanged)

score places value of the tag, rounded to the nearest integer, into the system variable "lscore"

score *expr* (value from 0 to 1000)

score (b) (sets "lscore" to -1)

System variables for routing

errtype = 0 for unknown error type
 = 1 for execution error
 = 2 for fatal condense error or for attempted jumpout to router
 not specified for course
 = 3 for memory exceeded
 = 4 for error in finish unit of instructional lesson
 = 5 for exit from condense queue via STOP1

ldone = -1 if student has encountered -lesson completed- or -end lesson-
 (and pressed NEXT)
 = 0 if student has encountered -lesson incomplete-
 = +1 if student has encountered -lesson noend-

lscore rounded value of tag of -score- (value from 0 to 100); equal to
 -1 if student has not previously worked lesson or if not specifically
 set by -score- or if set by -score (b)-; initially set to 0 for
 a student not routed by the system router

rcallow = 0 for no access to router common
 = 1 for -allow read-
 = 2 for -allow write-

router name of router lesson (left-justified)

rstartl name of lesson from last -restart- command (left-justified)

rstartu name of unit from last -restart- command (left-justified)

rvallow = 0 for no access to router variables
 = 1 for -allow read rvars-

SEQUENCING

Basic sequencing

unit names and initiates a section of a lesson (called a unit) which may be branched to by other sequencing commands

unit *name* (maximum of 8 characters in name; maximum length of unit is 500 condensed words)

Note: This command may have a form with arguments:

unit *name(var1,var2,var3,...)* (up to 10 arguments)

initial entry unit (ieu) refers to commands preceding first -unit- command in lesson; they are always executed whenever lesson is entered, no matter how (except for a router lesson)

entry equivalent to -goto *name*- followed immediately by -unit *name*-

entry *name*

NOTE: A lesson may have up to 395 different units referenced by -unit- and -entry-. No unit may be named "q" or "x".

Automatic sequencing

NOTE: The following commands (-jump-, -goto-, -do-, and -join-) may have conditional form, e.g.,

```
goto  expr,name0,name1,x,name3,q
do    expr,name0,name1,x,name3,q,var←initial,final,step
```

Argument x is equivalent to absence of the command; argument q is equivalent to a branch to a "blank" unit. Special case occurs with -do q- (join q-), which is equivalent to -goto q-. For iterative -do-, q terminates sequence, and x indicates no iteration is done for that value of conditional expression. Argument q is not valid with -jump-. Up to 100 arguments are permitted in the conditional tag.

These commands may have form with up to 10 arguments, e.g.,

```
goto  name(value1,value2,value3,,value5)  (values may be
:                                           expressions)
unit  name(var1,var2,var3,var4,var5)      (var4 is unchanged)
```

jump causes execution of unit named in tag with a full-screen erase (unless the erase is inhibited; see -inhibit erase-) and change of main unit; initializations associated with entering a main unit are performed

```
jump  name
```

goto causes execution of unit named in tag without screen erase and without change of main unit; since no main-unit initializations are performed, -goto- is milder than -jump-; there is no further execution of commands in the original unit except during the judging process

```
goto  name
```

do (insertion) causes execution of unit named in tag without screen erase or change of main unit before returning to original unit to execute commands following -do-

(iteration) causes repeated execution of unit(s) named in tag while changing a counter; otherwise same as insertion -do-

```
do    name
do    name,var←initial expr,final expr,stepsize expr (optional)
      (stepsize, if omitted, is +1; stepsize may be negative;
      loop variable is undefined after completion of the loop)
```

Note: Nested -do- and -join- levels may be up to 10 deep.

join similar to insertion -do- but will also be executed during judging and during search for additional -arrow- commands following an "ok" judgment

join *name*

exit permits termination of -do- or -join- sequences

exit (b) OR **exit** *negative value* (exit from all levels of -join- and -do-)
exit *expr giving num levels*
exit $\emptyset$ (causes no exit)

iferror specifies unit to -goto- if error is found in execution of a subsequent calculation in a unit

iferror *name*
iferror (b) OR **iferror** q (turns off iferror setting for remainder of unit)
iferror *expr, name₁, name₂, ..., name_n* (example of conditional form; maximum of 100 arguments in conditional tag)

imain unit named in tag is executed at start of every main unit in lesson; later occurrence of command overrides an earlier setting

imain *name*
imain (b) OR **imain** q (turns off imain setting for remainder of lesson or until reset)
imain *expr, name₁, name₂, ..., name_n* (example of conditional form; maximum of 100 arguments in conditional tag)

NOTE: The following two directives (-branch-, -doto-) are calc-type commands which permit branching or looping within a unit. When the directive is in the command field, it initiates a new -calc-. In the tag field, the directive is part of a continued -calc-. In both cases non-calculation commands are permitted within the -branch- or -doto- loop.

branch permits branching within a unit (statement label must start with a number and may contain up to 7 characters)

for example:

```

5a      var <= expr
:
:
branch  expr,5a,x,q  (argument x falls through to next line in
                    unit, argument q executes next non-calc
                    command)

calc    var <= expr
6test   var <= expr
branch  6test
branch  x           (falls through to next line in the -calc-)
branch  q           (executes next non-calc command)
branch  expr,q,x,6test,x  (example of conditional form)

```

doto permits looping within a unit (statement label must start with a number, may contain up to 7 characters, and must have a blank tag)

for example:

```

doto    2sync,var <= initial expr,final expr,stepsize expr
:
:
2sync   (b)

calc    var <= expr
doto    4 run,var <= initial expr,final expr,stepsize expr
:
:
4run    (b)

```

Note: Stepsize, if omitted, is +1. Stepsize may be negative. Value of loop variable is undefined after completion of the loop.

Key-initiated sequencing

NOTE: The following commands (-next-, -next1-, -back-, -back1-, -stop-, -nextnow-, -help-, -help1-, -data-, -data1-, -lab-, -lab1-, -helpop-, -help1op-, -dataop-, -data1op-, -labop-, -lab1op-) may have conditional form, where argument x leaves the pointer unchanged, and argument q clears the pointer and renders the key inactive (except for NEXT key, which will cause fall-through to following unit). Argument q is not valid with -nextnow-. Up to 100 arguments are permitted in the conditional tag. The conditional expression is evaluated when the command is executed, not when the key is pressed.

next, next1, back, back1, stop specifies unit executed when student presses appropriate key (arrows must be satisfied before branch on NEXT key)

```
next      name
back      name
back      (b) OR back    q  (clears back pointer; disables
                             BACK key)
```

nextnow terminates processing in unit and makes only the NEXT key active

```
nextnow name
```

help, help1, data, data1, lab, lab1 initiates help-type sequence by specifying unit to be executed if student presses appropriate key; sets base pointer for unit to return to unless base pointer is already set because this command is itself in a help-type sequence; new unit is a main unit but not a base unit (unless base pointer is cleared); help-type sequence can be terminated by -end-

```
help      name
lab       (b) OR lab    q  (clears lab pointer; disables LAB key)
```

helpop, help1op, dataop, data1op, labop, lab1op specifies unit executed when student presses appropriate key; new unit is not a main unit and no full-screen erase is executed (useful for on-the-page help)

```
helpop    name
dataop    (b) OR dataop q  (clears data pointer; disables
                             DATA key)
```

term permits use of the TERM key to initiate a help-type sequence starting at the unit containing this command and the specified character string; sequence can be terminated by -end- (see -inhibit term-)

term *string* (maximum of 8 characters)

base resets or clears base pointer in order to alter help-type sequencing

base *(b)* OR base *q* (clears base pointer)

base *name* (sets base pointer to named unit)

base *expr',q,x,name* (example of conditional form; argument *x* leaves base pointer unchanged; *q* clears base pointer; maximum of 100 arguments in conditional tag)

end terminates a help-type sequence or a lesson

end *(b)* OR end *help* (ends help-type sequence; ignored in non-help-type sequence)

end *lesson* (returns student to router lesson or to "Press NEXT to begin" page when NEXT is pressed; finish unit is not executed; returns authors to author-mode page)

Timing

group (non-executable) forms a group of keys with specified name for use with **-pause-** and **-keytype-** commands

group *name, key1, key2, key3, ...* (maximum of 7 characters in name)
group *name, name1, name2, ...* (groups may be combined)

used in:

pause *keys=key1, key2, name*
keytype *var, key0, key1, name, key3*

Note: System-defined groups are:

alpha (letters: a to z and A to Z)
numeric (digits: 0 to 9)
funct (function keys)
touch (input from touch panel)
ext (input from external device other than touch panel)
all (any keypress, touch-panel input, or external input)

pause delays execution of subsequent commands by specified interval or until specified keys are pressed

pause *expr giving num seconds* (minimum of .75 second)
pause *0* (causes no pause; exception to .75 second minimum)
pause *(b) OR pause negative value* (interrupts processing until any keypress comes in)
pause *keys=key1, key2, key3, ...* (interrupts processing until one of specified keys comes in; all keynames are typed in lower case [except for capitals] and without quotes)
pause *num seconds, keys=key1, key2, ...* (interrupts processing for specified time or until one of specified keys comes in)

Note: If a function key other than next, such as help, is specified and there is a preceding **-help-** or **-helpop-** command specifying a unit to execute, this unit is executed rather than the command following **-pause-**. If next is specified, the NEXT key just breaks the **-pause-**, even if there is a preceding **-next-** command.

keytype sets a variable according to relative position in a list of the key pressed by student; if key pressed is not listed, variable is set to -1

keytype *var, key0, key1, key2, ...*

time presses TIMEUP key after interval specified in tag and sets "key" to "timeup"

time *expr giving num seconds* (minimum of .75 second)
time (b) OR time *negative value* (turns off-time-in effect)

timer specifies unit in the router to which a routed student is sent when the indicated time has elapsed

timer *num seconds, name* (minimum of 300 seconds)
timer (b) (clears any -timer- in effect)

timel specifies a unit in the same lesson to execute (via a helpop-type branch) when the indicated time has elapsed; remains in effect across other timing commands and across unit boundaries

timel *num seconds, name* (minimum of 0.75 seconds)
timel (b) (clears any -timel- in effect)

press puts key named in tag into student input buffer for indicated station; limited to one keypress per second

press *keycode*
press "keyname" (for non-function keys)
press *keyname* (for function keys, e.g., press next)
press *keycode, station* (presses key at a different station if that station is in the same lesson as -press-; "zreturn" is set to -1 if station in lesson, to 0 if not)

catchup (no tag) causes pause in execution while transmission to the terminal of accumulated output is completed in order to synchronize display and execution of commands

return (no tag) interrupts processing and returns with new time slice for further processing

cpulim tag sets the maximum CPU usage rate in milliseconds per second up to 100 milliseconds per second

cpulim *expr* (value of tag from 1 to 100)

Lesson connections and sections

`cstart` (non-executable) (no tag) indicates following code is to be condensed (used after a preceding `-cstop-`)

`cstop` (non-executable) (no tag) indicates following code is not to be condensed and is effective up to the next `-cstart-`, if any

`cstop*` (non-executable) (no tag) indicates none of the following code is to be condensed, independent of subsequent `-cstart-` commands

NOTE: It is preferable to use the partial condense option of the editor, rather than `-cstart-`, `-cstop-`, and `-cstop*-`.

`use` (non-executable) inserts specified block(s) from the lesson specified on the lesson information page into the lesson being condensed; all contiguous blocks with the same name are taken; use codes on lessons must match

`use` *block name*

`jumpout` causes execution of commands in lesson named in tag, in specific unit in that lesson, if given, otherwise in first unit (see `-inhibit jumpchk-` and `-inhibit from-`)

`jumpout lesson name` (goes to first unit in lesson; jumpout codes need not match)

`jumpout lesson name,unit name` (jumpout codes must match)

`jumpout resume` (allows router lesson to return student to lesson and unit specified by last `-restart-`)

`jumpout return` (returns to first unit of lesson from which a jumpout was made to present lesson)

`jumpout return,return` (returns to lesson from which jumpout was made to unit following the unit which contains the `-jumpout-`)

`jumpout (b)` OR `jumpout q` (causes jumpout to author-mode page for authors and to "Press NEXT to begin" page or to router for students; similar to `-end lesson-`)

`jumpout <leslist position>` (causes jumpout to first unit of lesson in specified position in leslist)

`jumpout <leslist position>,unit name` (causes jumpout to specified unit in lesson in leslist position indicated; variable unit name must be enclosed in parentheses)

`jumpout expr;lessonm,unitm;lessonl;lessonl,unitl;q;x` (example of conditional form; argument `q` causes jumpout as above; argument `x` causes no jumpout)

from checks the lesson and main unit, if specified, from which a lesson was entered against a list, and sets a variable to the relative position of the lesson and unit in the list; if the lesson and unit are not found, the variable is set to -1 (if no unit is specified, any unit in the lesson qualifies)

for example:

```
from    var; lesson0, unit0; lesson1; lesson2, unit2;
        <leslist position>, unit3
```

lessin checks if the lesson specified in the tag is credited to the user's logical site; sets system variable "zreturn" to -1 if the lesson is in ECS and in use at the user's site and to 0 if the lesson is not in use at the user's site; used to check whether a lesson is available for immediate use with -jumpout-

```
lessin  'lesson name'
lessin  (var containing lesson name)
lessin  <leslist position>
```

in sets system variable "zreturn" to indicate whether the station number specified is in the lesson containing the -in- command

```
in      expr    (value of tag is rounded to integer from 0 to 1023)
```

```
zreturn = -2 if -in- command is in router lesson and routed student
           at specified station is in instructional lesson
          = -1 if station is in instructional lesson containing -in-
          = 0  if station is not in the lesson
```

Lesson Lists

leslist references two special blocks containing a list of up to 322 lessons (numbered 0 to 321) available for general use; common codes on the lesson containing the leslist blocks and the lesson containing -leslist- command must match

leslist *lesson name, block(s) name*

Note: zreturn = -1 if the -leslist command is executed successfully
 = 0 if the leslist is not found
 = 1 if the common codewords do not match
 = 2 if this is a duplicate leslist reference

addlst allows addition of lesson names to a leslist either in the specified slot or in the first empty slot if none is specified without editing the leslist directly; i.e., names are added from student mode; tag must be a variable; requires three consecutive variables containing information on lesson names (names are stored with: -storea *starting var, 30-*)

addlst *starting var, leslist position (optional)*

Note: zreturn = -1 if the lesson name is added successfully
 = 0 if there is no preceding successful -leslist- command
 = 1 if the form of the lesson name is incorrect
 = 2 if the lesson name is already in the leslist
 = 3 if the leslist is full
 = 4 if the specified slot is in use (with the two-argument form)

removl allows deletion of a lesson in a specified position in a leslist from student mode (similar to -addlst-); vacated position is left blank

removl *leslist position*

Note: zreturn = -1 if the lesson is removed successfully
 = 0 if there is no preceding successful -leslist- command

`lname` stores the lesson name in the specified `leslist` position in three consecutive variables starting at the specified variable

`lname` *starting var, leslist position*

Note: Use with `-showa-` to display the lesson name; e.g.:

`lname` *starting var, leslist position*
`showa` *starting var, 30*

`zreturn` = -1 if execution is successful
 = 0 if there is no preceding successful `-leslist-` command

`findl` searches the `leslist` for the lesson name stored in three consecutive variables and returns the `leslist` position in the specified variable

`findl` *starting var for lesson name, var for leslist position*

Note: If the lesson name is not found or if no `leslist` is used, returned value for the `leslist` position = -1

Lesson annotation and debugging

* indicates statement on that line is a comment only and is to be
 c(space) ignored by the computer

*This is a comment.
 c This is a comment.

\$\$ (not a command) when placed on same line with a TUTOR statement,
 indicates that subsequent material on that line is a comment

command tag \$\$This is a comment.

change (non-executable) permits names of commands to be changed, e.g.,
 to a foreign language

change *normal TUTOR name to new name*

for example:

change at to wo

step allows author to step through lesson command by command; also
 available via TERM step (not available to students)

step on OR step *expr* $\neq \emptyset$ (turns on step option)

step off OR step *expr* $= \emptyset$ (turns off step option)

See next page for *list options.

- *list** (pseudo-command not affecting condensing or execution) allows certain print options for listing a lesson
- *list charset** (prints contents of charset with 0 for dots on and - for dots off)
 - *list charset,(dotsblank)** (prints contents of charset with symbols specified for dots and blanks)
 - *list commands,command1,command2,command3,...** (up to 10 commands; lists lines on print where specified commands appear)
 - *list common,block name,num words,format** (prints contents of common; see Note below for information on format)
 - *list dataset,starting block number,num words,format** (prints contents of dataset; see Note below for information on format)
 - *list eject** (causes page eject where command is located)
 - *list ignore** (causes all subsequent **-*list-** commands to be ignored)
 - *list info** (prints lesson information display)
 - *list label,your label including spaces** (prints label at location of command)
 - *list off** (stops printing source code at location of command and starts printing unit cross-reference table and any preceding ***list** options)
 - *list off,blockname1,blockname2,blockname3-blockname4** (specifies blocks that are not to be printed)
 - *list parts** (prints only those blocks which have condense flag turned on)
 - *list special** (same as printing dataset with special format except that print always starts at beginning of dataset; see Note below for special format options)
 - *list symbols** (prints cross reference of variables, defined and primitive, used in the lesson)
 - *list text** (prints only text of **-write-** and **-writec-** commands)
 - *list title,your title** (specifies subheading to be printed under lesson name on each page)
 - *list varian,charset,lesson name,charset name** (prints lesson using the charset in specified lesson on varian printer)
 - *list vocab** (prints contents of subsequent vocab blocks)

Note: For `-*list common-` and `-*list dataset-` format may be:

integer	or i	(nc-variables, 10 words per line)
exponential	or e	(vc-variables, 10 words per line)
octal	or o	(5 words per line)
alpha	or a	(10 words per line)
x		for all formats (2 words per line)
special	or s	for special print (specified number of words to be printed is ignored since entire blocks are printed; see details below)
(designed)		specially designed format (enclosed in parentheses; see details below)

special format:

words are interpreted in alpha and words identical to 0 are ignored;
up to 120 characters are printed per line;
last word on the line must end in 12 bits of 0

print options available are:

*format eject	(causes page eject at location of this directive in the common or dataset)
*format blocks	(signals page eject after each block; placed at beginning of common or dataset)
*format end	(indicates end of print at location of directive in the common or dataset)

(Each of these directives requires two consecutive words in the common or dataset.)

designed format:

format must be enclosed in parentheses;
up to 135 characters are printed per line;

formats available:

i	for integer	(nc-variables, 10 characters per word)
e	for exponential	(vc-variables, 10 characters per word)
o	for octal	(20 characters per word)
a	for alpha	(10 characters per word)
x	for space	(preceding number indicates number of spaces)
l	for common or dataset location	(4 characters)
p	for skip to next word to be printed on same line	(preceding number indicates how many words to go forward)

System variables for sequencing

args	number of arguments transferred at the previous execution of a unit with arguments
backout	= 0 if the student signs off via STOP1 ≠ 0 if the student exits via system backout
baseu	name of the student's current base unit (left-justified); = 0 if no base unit is specified, indicating the student is not in a help-type sequence
clock	value of the system clock in seconds to the nearest millisecond since the previous deadstart (see also command -clock-)
fromnum	leslist position of the lesson from which the student came via -jumpout-; = -1 if the lesson is not in a leslist or if no leslist is being used
key	contains 7-bit keycode of last keypress; after a touch-panel input contains a 9-bit number which gives the location of the touch square touched; the binary form of this number is: 1xxxxyyy, where the 4 bits labeled "x" give the horizontal touch location and the 4 bits labeled "y" give the vertical touch location; "key" is useful with -pause-, -time-, and -arrow-
lessnum	leslist position of the student's current lesson; = -1 if the lesson is not in the leslist or if no leslist is being used
llesson	length of lesson
mainu	name of the student's current main unit (left-justified)
mallot	memory allotment for the logical site at which the user is working
muse	total memory usage by users at the same logical site as the user
nhelpop	number of times a help-type key is pressed for on-the-page help; zeroed for each main unit and for each arrow in the unit

proctim CPU usage in seconds (accurate to 1 millisecond)

p_time = -1 if current time is during prime-time hours
 = 0 otherwise

sitenam name of the user's logical site (left-justified)

station identification number assigned by the system to a terminal
 physical site = station \$ars\$ 5
 site-station # = station \$mask\$ 037
 station = 32 × physical site + site-station #

tactive number of currently active terminals

user user type: author, instructor, student, multiple, sabot (if
 student records have been aborted), snockpt (if automatic check-
 point has been aborted) (left-justified)

usersin number of students in the lesson (routed students are counted as
 being in the router as well as in the instructional lesson)

zreturn general-purpose system variable--set with certain commands
 according to the results of an operation (See -addlst-, -charset-,
 -chartst-, -datain-, -dataout-, -dataset-, -ext-, -in-, -leslist-,
 -lessin-, -lname-, -micro-, -press-, -readd-, -readr-, -readset-,
 -release-, -removl-, -reserve-)

zwpb number of words per block in a lesson (currently 322)

NOTE: "baseu", "mainu", "sitenam", and "user" must be displayed with -showa-.

APPENDIX

Some limits associated with commands

area	10 characters in name of area
argumented unit	10 arguments
bump	8 characters
calcc calcs	61 calculations
common (permanent)	8050 variables (>1500 variables requires -comload-)
common (temporary)	1500 variables
comload	1500 variables
compute	99 characters in character string
conditional form of commands which reference unit names, e.g., next, help, etc.	100 unit names (line containing 101st name is flagged as condense error)
course	8 characters in sign-on course name
cpulim	value of tag from 1 to 10
define	7 characters in name of define set 7 characters in name of variable 400 definitions (or less if definitions are long) 6 arguments in defined function 10 units (e.g., kg, m, sec, liter, etc.)
delay	1 second maximum
do } join }	combined 10 levels
draw rdraw	63 arguments
endings	10 separate -endings- commands, 7 endings in each tag
graph	9 characters in string to be plotted

group	7 characters in name
hbar	9 characters in string to be plotted
vbar	
in	value of tag from 0 to 1023
list	7 characters in name of list
long	300 characters (>150 requires -edit- for active EDIT key)
move	5000 characters may be moved
name	18 characters in user's sign-on name
noword	9 characters in word
okword	
outputl	10 characters in label, 20 consecutive variables
pause	.75 second minimum
put	50 characters in strings; expansion due to string replacement limited to 300 characters
putd	
putv	
routvar	50 variables
score	value of tag from 0 to 100
storage	1500 variables
term	8 characters in string
time	.75 second minimum
timel	.75 second minimum
timer	300 seconds minimum
transfr	length is the smaller of: size of common or storage (reference to ECS) OR length of -comload- or -stoload- (reference to CM) OR 150 (reference to student variables) OR tag of -routvar- (reference to router variables)
unit	} combined
entry	
	8 characters in name, 500 condensed words in unit, 395 distinct condensed units in lesson
vocab	7 characters in name of vocabulary
vocabs	
*list commands	10 commands

Alphabetical index to system variables

These system variables have been classified (in some cases, approximately) according to their functions. They may be used wherever expressions are accepted, e.g., in tag of -calc-, -at-, etc.

Word	Page	Word	Page	Word	Page
aarea	D7	fromnum	S16	router	R3
aarrows	D7	jcount	J16	rstartl	R3
ahelp	D7	judged	J16	rstartu	R3
ahelpn	D7	key	S16	rvallo	R3
anscnt	J16	lcommon	C17	sitenam	S17
ansok	J16	ldone	R3	size	P16
aok	D7	lessnum	S16	spell	J17
aokist	D7	llesson	S16	station	S17
args	S16	lscore	R3	tactive	S17
asno	D7	lstorag	C17	user	S17
aterm	D7	mainu	S16	usersin	S17
atermn	D7	mallot	S16	varcnt	J17
atime	D7	mode	P16	wcount	J17
auno	D7	muse	S16	where	P16
backout	S16	nhelpop	S16	wherex	P16
baseu	S16	ntries	J16	wherey	P16
capital	J16	opcnt	J17	zbpc	C17
clock	S16	order	J17	zbpw	C17
dataon	D7	phrase	J17	zcpw	C17
entire	J16	proctim	S17	zdsrecs	C17
errtype	R3	ptime	S17	zdsrpr	C17
extra	J17	rcallow	R3	zreturn	S17
formok	J17			zwpb	S17

Alphabetical index to commands and related directives

Command	Page	Command	Page	Command	Page	Command	Page
abort	C12	char	P4	delta	P12	gdraw	P10
addlst	S11	charset	P14	disable	P13	goto	S2
add1	C2	chartst	P14	do	S2	graph	P10
allow	R1	circle	P6	dot	P6	group	S7
altfont	P15	circleb	P7	doto	S4	hbar	P11
ans	J11	clock	C8	draw	P6	help	S5
ansu	J10	close	J4	edit	J2	help1	S5
ansv	J10	color	P3	embed	P1	helpop	S5
answer	J8	comload	C12	enable	P13	help1op	S5
answerc	J9	common	C12	end	S6	iarrow	J13
array	C1	compute	C8	endarrow	J2	ieu	S1
area	D2	concept	J9	endings	J3	iferror	S3
argument	S1, S2	copy	J2	entry	S1	ignore	J12
arrow	J2	course	C8	erase	P3	imain	S3
at	P1	cpulim	S8	eraseu	J1	in	S10
atnm	P1	cstart	S9	exact	J9	inhibit	P4
audio	P13	cstop	S9	exactc	J9	initial	C13
axes	P8	cstop*	S9	exactv	J10	itoa	C10
back	S5	data	S5	exit	S3	jkey	J2
back1	S5	data1	S5	ext	P13	join	J13, S3
backgnd	C16	datain	C14	find	C9	judge	J14
base	S6	dataoff	D1	findall	C9	jump	S2
block	C11	dataon	D1	find1	S12	jumpout	S9
bounds	P8	dataop	S5	finish	D6	keytype	S7
branch	S4	data1op	S5	force	J1	lab	S5
bump	J4	dataout	C14	foregnd	C13	lab1	S5
c	S13	dataset	C14	frame	P8	labelx	P9
calc	C2	date	C8	from	S10	labeledy	P10
calcc	C2	day	C8	funct	P12	labop	S5
calcs	C2	define	C1			lab1op	S5
catchup	S8	delay	P4				
change	S13						

Command	Page	Command	Page	Command	Page	Command	Page
leslist	S11	pack	C9	showa	P3	window	P7
lessin	S10	pause	S7	showe	P2	write	P1
lesson	R2	play	P13	showo	P2	writec	P1
lineset	P15	plot	P5	showt	P2	wrong	J8
list	J3	polar	P11	showz	P2	wrongc	J9
lname	S12	press	S8	size	P3	wrongu	J10
loada	J4	put	J4	slide	P13	wrongv	J10
locate	P10	putd	J4	specs	J5	zero	C3
long	J2	putv	J4	step	S13	*	S13
lscalex	P9	randp	C7	stoload	C13	\$\$	S13
lscaley	P9	randu	C7	stop	S5	*format	S15
markx	P10	rdraw	P6	storage	C13	*list	S14
marky	P10	readd	D3	store	J7		
match	J8	readr	D4	storea	J7		
micro	P14	readset	D3	storen	J7		
miscon	J9	record	P13	storeu	J7		
mode	P3	release	C15	sub1	C3		
modperm	C7	remove	C7	tabset	P14		
move	C10	removl	S11	term	S6		
name	C8	reserve	C15	time	S8		
next	S5	restart	D6	timel	S8		
next1	S5	return	S8	timer	S8		
nextnow	S5	rotate	P3	touch	J11		
no	J12	route	R1	touchw	J11		
noword	J15	routvar	R1	transfr	C11		
ok	J12	scalex	P8	unit	S1		
okword	J15	scaley	P9	use	S9		
open	J7	score	R2	vbar	P11		
or	J11	search	C10	vector	P11		
origin	P8	segment	C1	vocab	J3		
output	D2	set	C3	vocabs	J3		
outputl	D2	setperm	C7				
		show	P2				

END

GUIDE

Standard PLATO Keyset

Upper Case	Micro, Upper Case
Lower Case	Micro, Lower Case

< 0	≤ 1	≥ 2	[{	] }	\$ #	% 5	6	7	*	(9	≡ ≠	SUPER	SUB	TERM ANS	COPY
CR TAB	Σ +	Q q	W w	E e	R r	T t	Y y	U u	I i	O o	P p	ERASE	FONT MICRO	HELP	□
←	Δ -	A a	S s	D d	F f	G g	H h	J j	K k	L l	:	NEXT	EDIT	BACK	LAB
C +	X °	Z z	X x	C c	V v	B b	N n	M m	" "	! .	? /	SHIFT	DATA	STOP	

BACK SPACE
SPACE